

Pandora reconstruction tutorial

L. Escudero
for the Pandora Team

ProtoDUNE Analysis Workshop
27th January 2019





Introduction



Earlier today:

- You have seen a summary of Pandora and its reconstruction for ProtoDUNE, in John's talk [here](#).

Now:

- We will go through some practical information about using Pandora, with a couple of small exercises.
- We won't have much time for hands-on practice today, but the exercises here and in backup should be ready for you to follow them, and to have a look at other sources linked in these pages.

Later:

All our code is Open Source and available in GitHub

<https://github.com/PandoraPFA>

Lots of documentation and workshop tutorials available (from using output to write your own algorithms)

- Pandora MicroBooNE workshop in 2016, material [here](#)
- UK (Manchester) LArSoft workshop 2018 exercises [here](#)
- Workshop in Paraguay 2018 exercises [here](#)

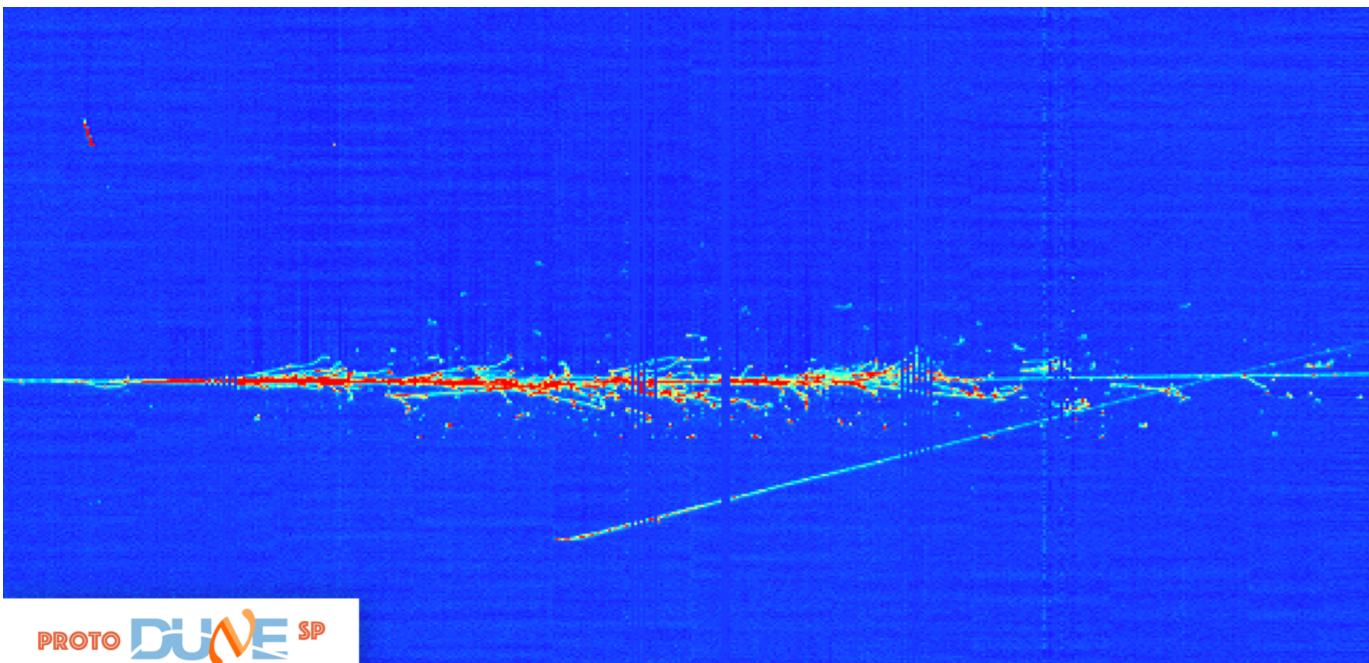


Introduction

DUNE

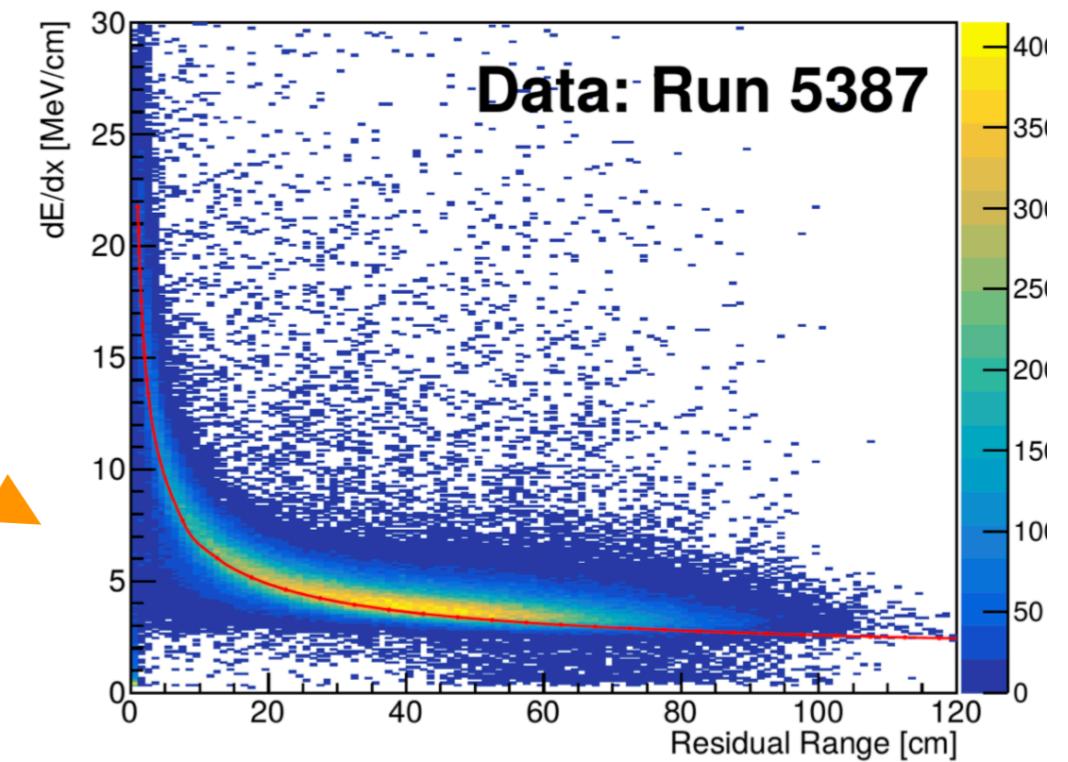
Welcome to your journey to physics in ProtoDUNE!

Going from here...



ProtoDUNE data event, Flavio Cavanna, W&C seminar (Sep18)

...to here...



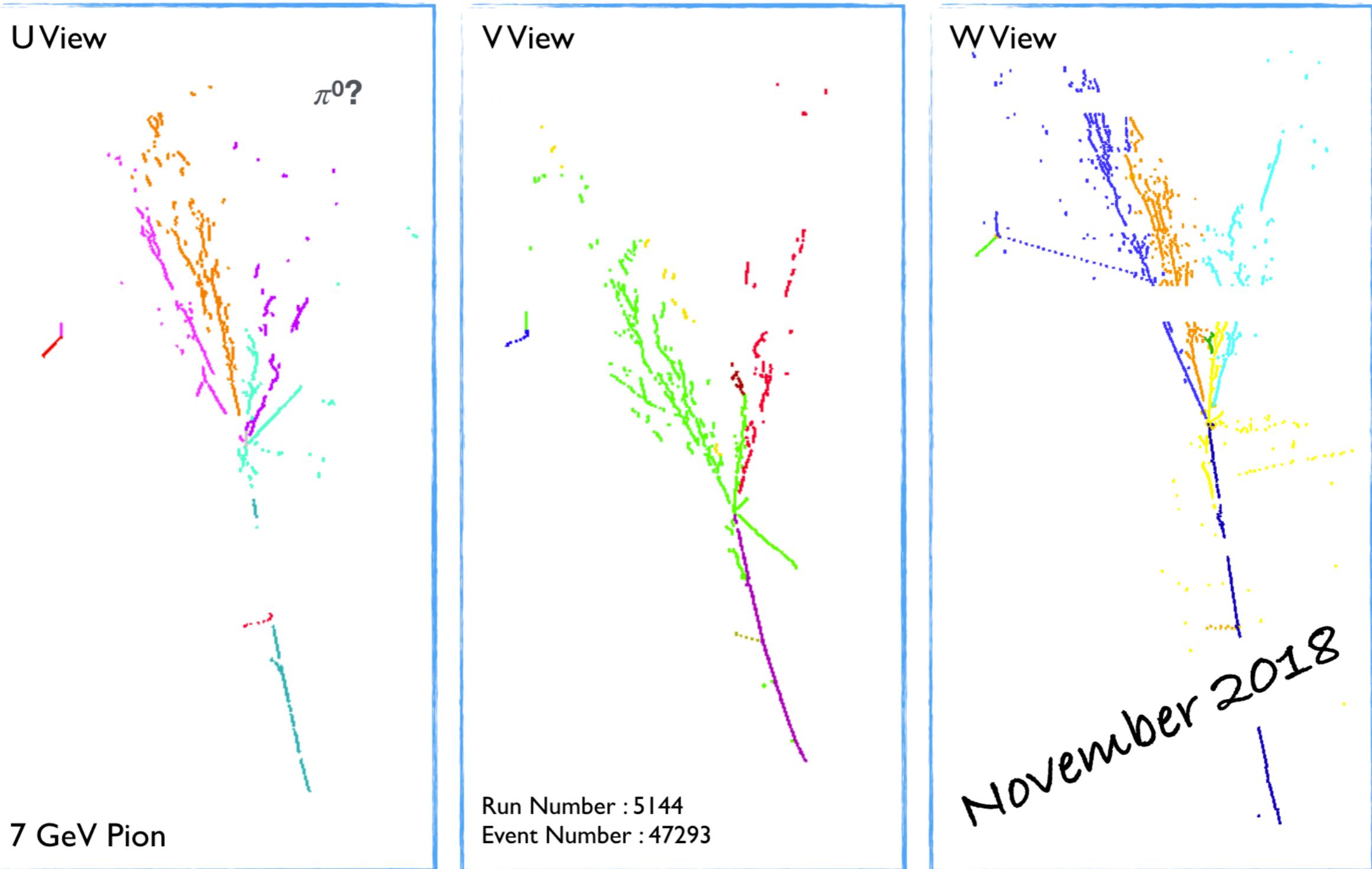
Heng-Ye Liao, ProtoDUNE Sim/Reco meeting 23rd January



Introduction

DUNE

Welcome to your journey to physics in ProtoDUNE!



Please enjoy the trip



I) Running the reconstruction and recognising the output



Running the reconstruction



Example:

```
lar -c protoDUNE_reco.fcl -n 10 my_simulated_events.root
```

Note:

- You can find some detsim events in: </dune/app/users/lorena/ProtoDUNEWorkshop/>
- Or use any available files, have a look at the samweb definitions here: <http://dune-data.fnal.gov>
- or see backup to generate some yourself

See backup for more details on what is happening, what's inside the .fcl file, and a small exercise

```
reco: [ rns,
    #optical hits and flashes
    ophit, opflash,
    #TPC wire signals
    caldata,
    #hit reconstruction
    gausshit, #fasthit,
    #space point solver
    reco3d,
    #real disambiguation
    hitpdune
    #cluster reco
    linecluster,
    #feature labeling
    emtrkmichelid,
    #pandora
    pandora, pandoraTrack, pandoraShower,
    pandoracalo, pandorapid, pandoracali, pandoracalipid,
    #pmtrack
    pmtrack, pmtrackcalo, pmtrackpid, pmtrackcali, pmtrackcalipid
    #pmtrajfit
    pmtrajfitcalo, pmtrajfitpid]
```

List of processes run in the reconstruction stage (more info in backup)

These are the Pandora producers

Important: things downstream of Pandora sometimes contain its name if they use Pandora's output, but these modules are NOT part of Pandora

Reminder of LArSoft syntax:

```
lar -c fihcl_file_to_run.fcl -n number_of_events(optional)
input_file_name.root [-o output_file_name.root (optional)]
```



Pandora producers and output

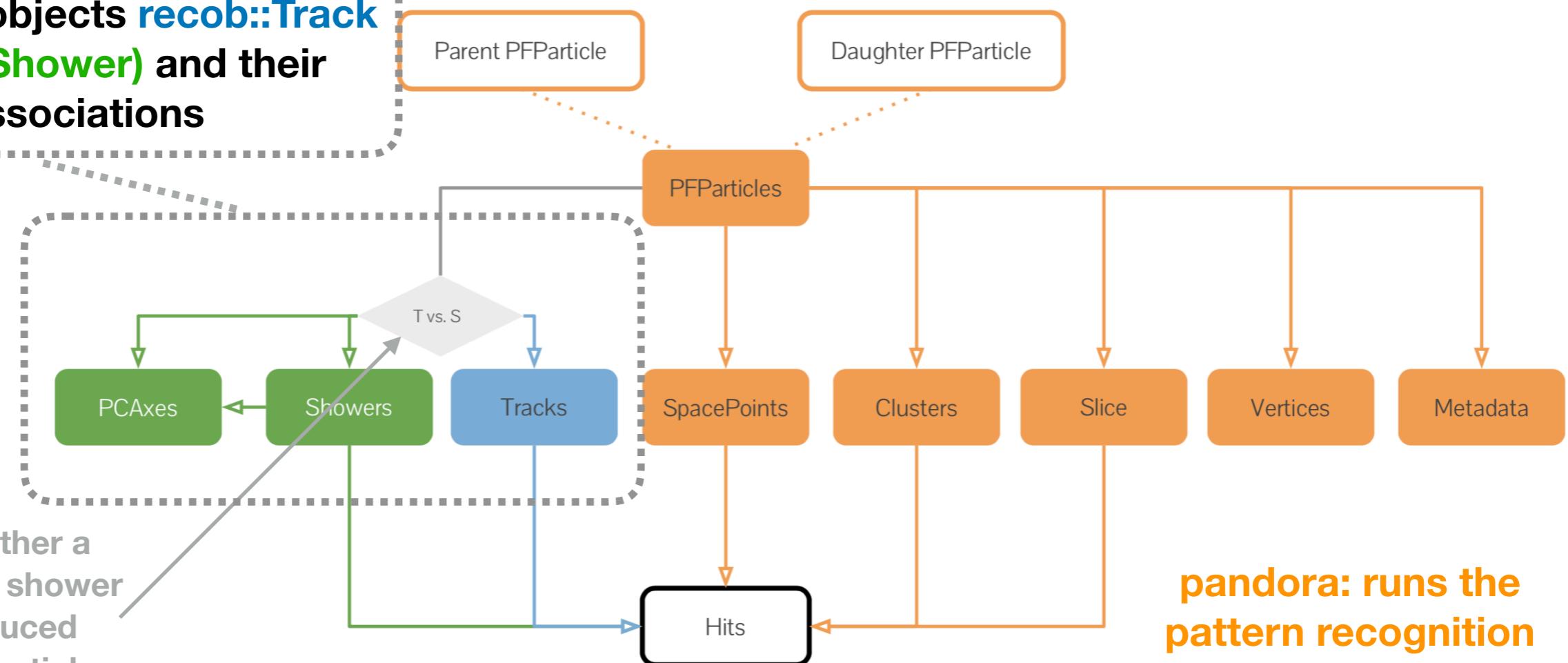
DUNE

```
pandora:          @local::protodune_pandora
pandoraTrack:     @local::dune  pandoraTrackCreation
pandoraShower:    @local::dune  pandoraShowerCreation
```

See backup
for more info

pandoraTrack (pandoraShower):
LArSoft module producing high level reco objects `recob::Track` (`recob::Shower`) and their associations

Pandora Output to LArSoft





Pandora producers and output

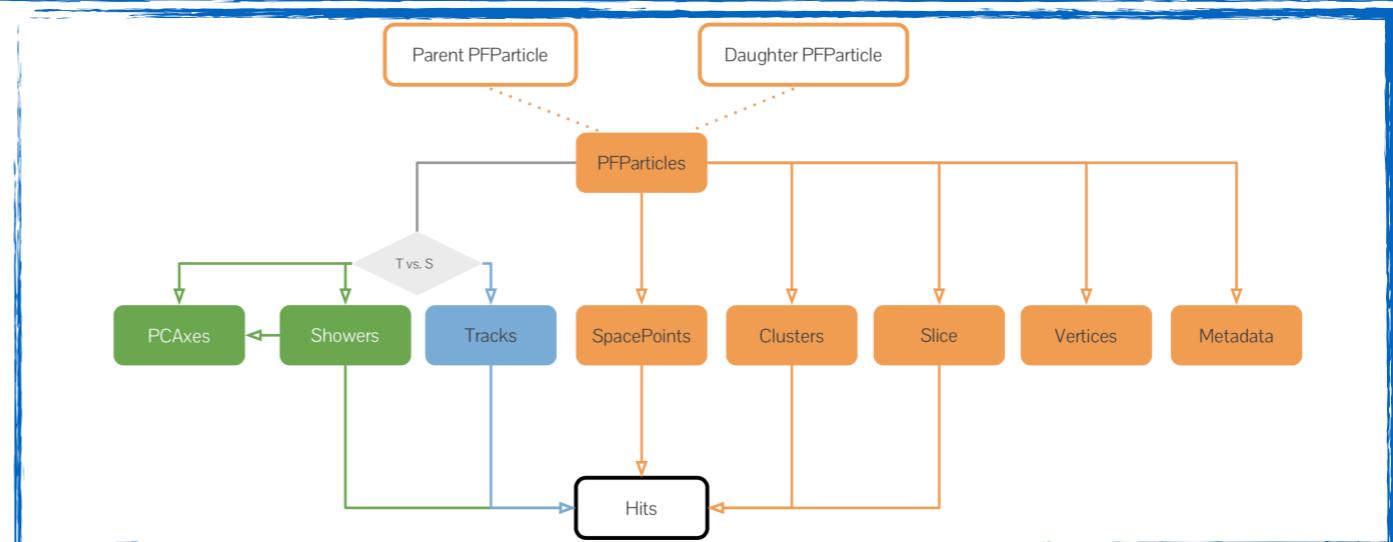


Pandora products

PROCESS NAME	MODULE_LABEL..	PRODUCT INSTANCE NAME..	DATA PRODUCT TYPE.....	.SIZE
Reco.....	pandora.....	std::vector<recob::PFParticle>.....	... 12
Reco.....	pandora.....	std::vector<recob::Vertex>.....	... 11
Reco.....	pandora.....	std::vector<larpandoraobj::PFParticleMetadata>.....	... 12
Reco.....	pandora.....	std::vector<recob::SpacePoint>.....	.. 508
Reco.....	pandora.....	std::vector<recob::Cluster>.....	... 3
Reco.....	pandora.....	std::vector<anab::T0>.....	... 0
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::SpacePoint, void>.....	.. 508
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::Vertex, void>.....	... 11
Reco.....	pandora.....	art::Assns<recob::Cluster, recob::Hit, void>.....	.. 620
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::Cluster, void>.....	... 33
Reco.....	pandora.....	art::Assns<recob::PFParticle, larpandoraobj::PFParticleMetadata, void>.....	... 12
Reco.....	pandora.....	art::Assns<recob::PFParticle, anab::T0, void>.....	... 0
Reco.....	pandora.....	art::Assns<recob::SpacePoint, recob::Hit, void>.....	.. 508
Reco.....	pandoraTrack..	std::vector<recob::Track>.....	... 7
Reco.....	pandoraTrack..	art::Assns<recob::PFParticle, recob::Track, void>.....	... 7
Reco.....	pandoraTrack..	art::Assns<recob::Track, recob::Hit, void>.....	.. 345
Reco.....	pandoraTrack..	art::Assns<recob::Track, recob::Hit, recob::TrackHitMeta>.....	.. 345
Reco.....	pandoraShower..	std::vector<recob::Shower>.....	... 4
Reco.....	pandoraShower..	std::vector<recob::PCAxis>.....	... 4
Reco.....	pandoraShower..	art::Assns<recob::Shower, recob::Hit, void>.....	.. 257
Reco.....	pandoraShower..	art::Assns<recob::PFParticle, recob::PCAxis, void>.....	... 4
Reco.....	pandoraShower..	art::Assns<recob::PFParticle, recob::Shower, void>.....	... 4
Reco.....	pandoraShower..	art::Assns<recob::Shower, recob::PCAxis, void>.....	... 4

lar -c eventdump.fcl my_reco_events.root

See more in backup

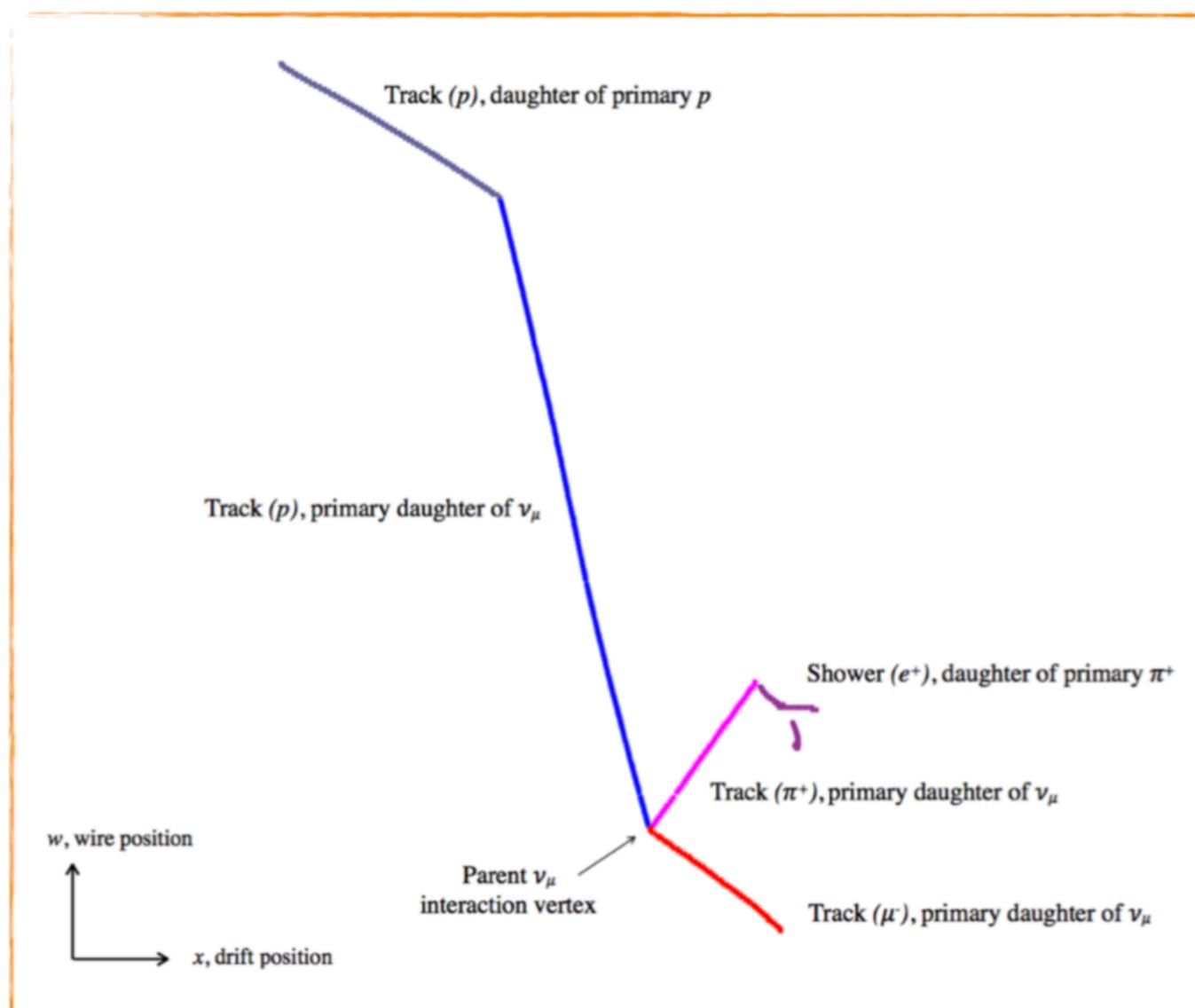




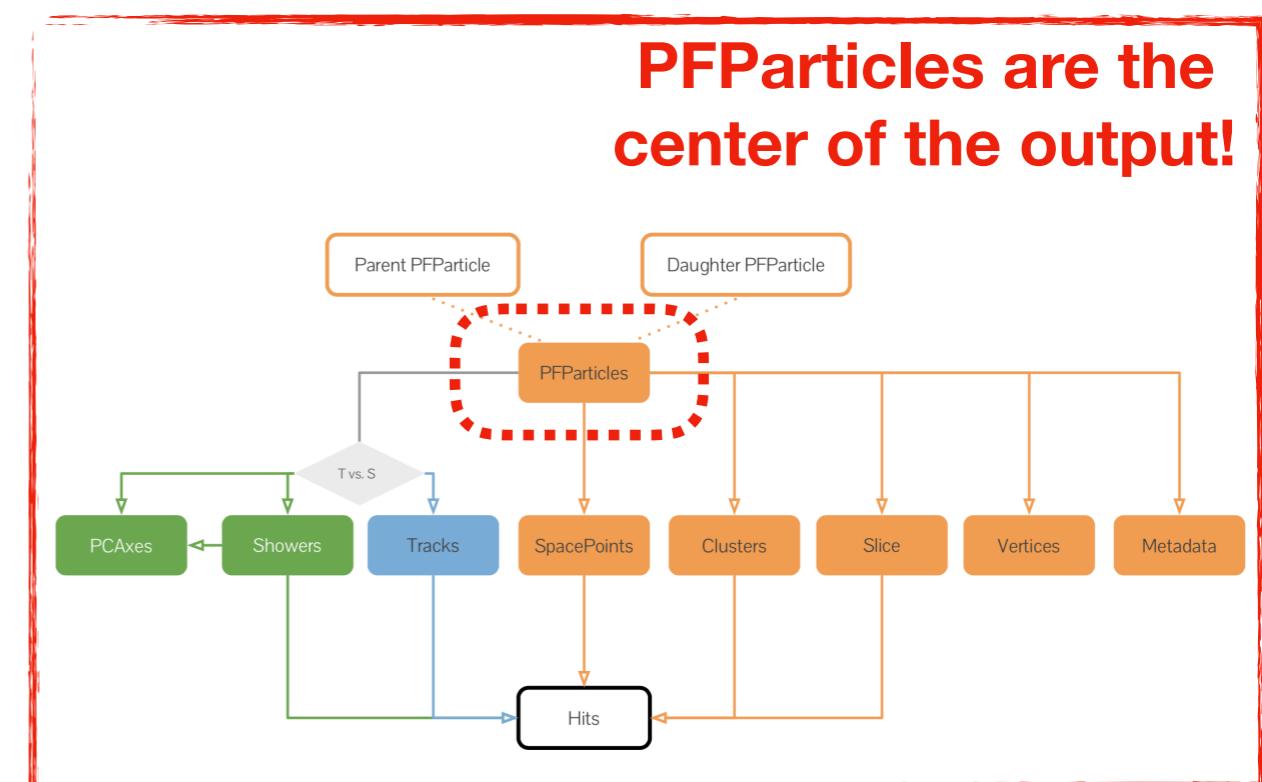
PFParticles & Tracks/Showers

DUNE

The main output of Pandora is a set of **PFParticles** (Particle Flow Particles)
They represent 3D reconstructed particles, with their associated objects
(Clusters, SpacePoints, Vertices, etc)



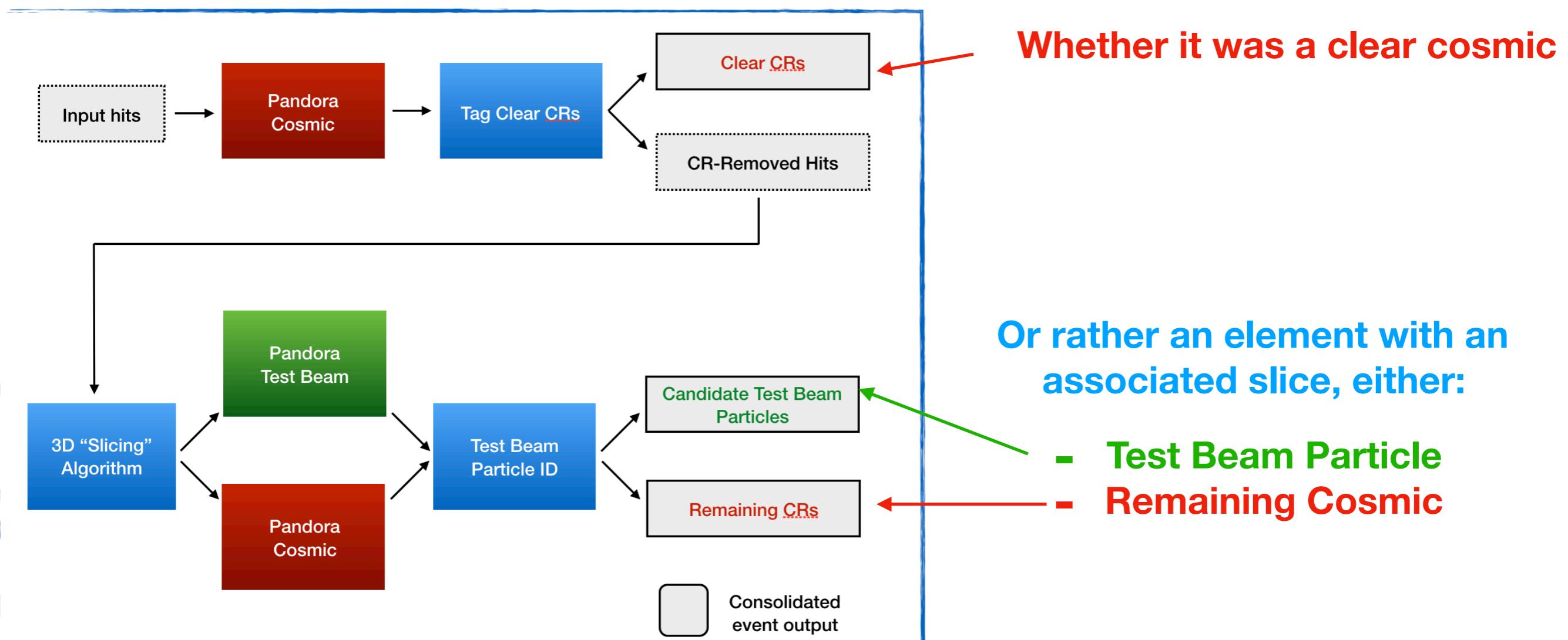
Each PFParticle is classified as track-like or shower-like (according to topological features), and the corresponding object is created



They are organised into hierarchies (that describe the interaction flow) and can be navigated via parent-daughter links

Remember to use PFParticles as well as Tracks and Showers, to understand the hierarchy occurred!

PFParticles also have associated metadata, informing of the type of reconstruction they are output of:



This will be stored in the metadata with a string (e.g. `SliceIndex`, etc) - we will see examples later



2) Visualising events



Visualise events

Pandora Visualisation uses ROOT TEVE, which usually works nicely on most linux and Mac systems, but can be a bit slow embedded in a complex application such as LArSoft.

For the visualisation to run, you need to have an installation of ROOT including the TEVE libraries. You can test the libEve functionality by attempting to run the ROOT tutorial: tutorials/eve/calorimeters.C

So... don't be frustrated if you try to run it and it crashes
- check your GLX (openGL X11 forwarding)!

Today, lets just connect to one of the dunegpvm nodes

Don't forget to ssh with -X option

And set up your environment (see backup if needed)

And be patient! Running the visualisation inside LArSoft and x forwarding is slow!



Visualise events

But you have another alternative (thanks Leigh and Dom!) through VNC:

Follow instructions here:

[https://wiki.dunescience.org/wiki/DUNE_Computing/
Using_VNC_Connections_on_the_dunegpvms](https://wiki.dunescience.org/wiki/DUNE_Computing/Using_VNC_Connections_on_the_dunegpvms)

This will be much faster!

From one of the gpvm machines:

```
vncserver :XX -localhost  
export DISPLAY=localhost:XX
```

DO NOT FORGET -LOCALHOST

XX a number of your choice

From a local terminal:

```
ssh -K -L 59XX:localhost:59XX -N -f -I USERNAME dunegpvmYY.fnal.gov  
open vnc://localhost:59XX (for MAC)
```

Run the command from the gpvm machine (next slides)



Visualise events

The Pandora algorithms selected to run in each configuration (e.g. cosmic vs neutrino/test beam, DUNEFD vs protoDUNE) are specified by .xml settings files

Here it is specified which .xml settings file will be used:

Inside `$DUNETPC_DIR/job/pandoramodules_dune.fcl`:

```
dunefd_pandora:  
dunefd_pandora.ConfigFile:  
dunefd_pandora.ShouldRunNeutrinoReco0ption:
```

```
@local::dune_pandora  
"PandoraSettings_Master_DUNEFD.xml"  
true
```

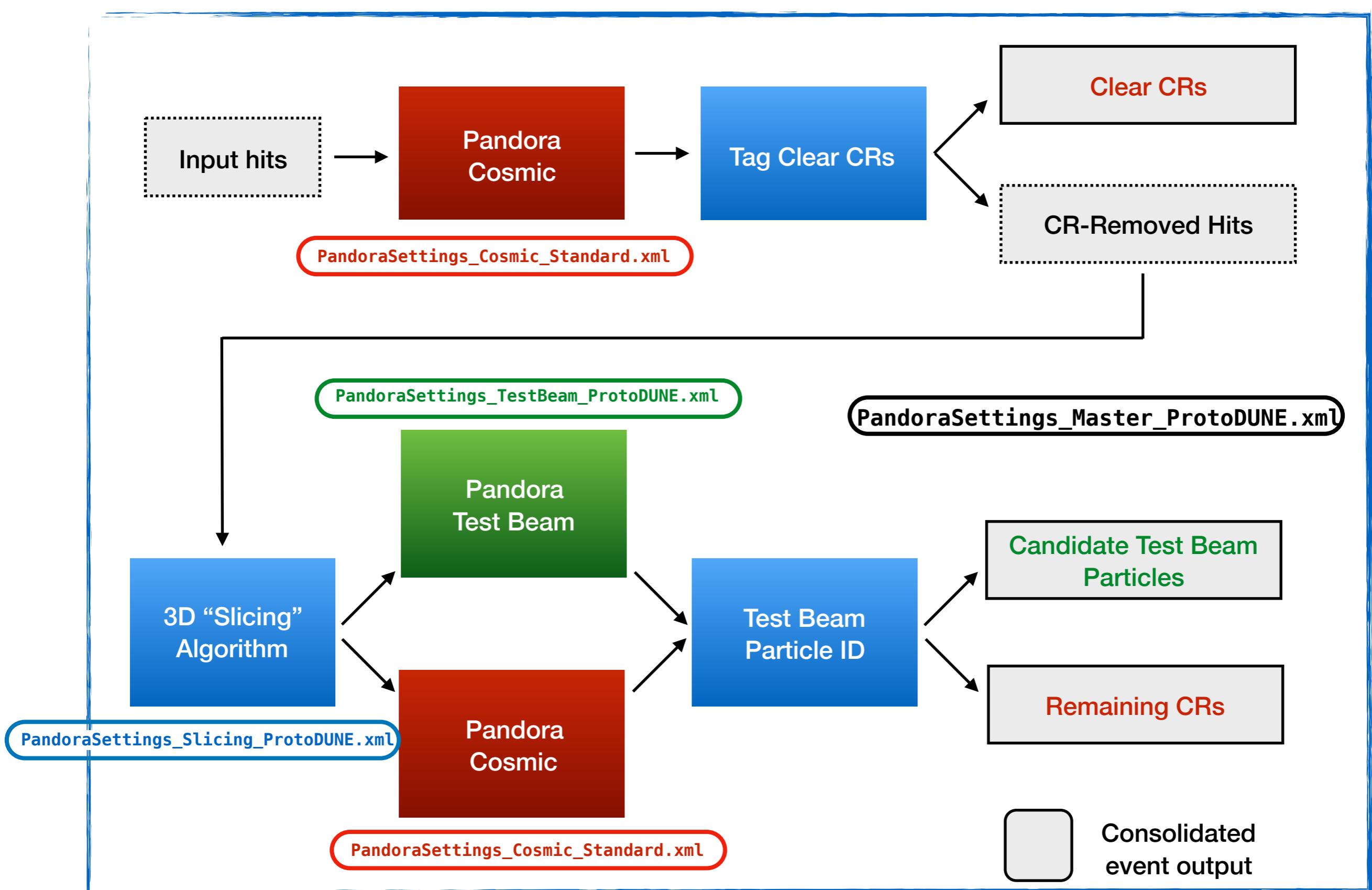
```
protodune_pandora:  
protodune_pandora.ConfigFile:
```

```
@local::dune_pandora  
"PandoraSettings_Master_ProtoDUNE.xml"
```

These are the master settings files which control the flow of algorithms specified in other settings files to provide a consolidated output



Consolidated Output





Visualise events

Let's make a copy of the master settings file and look inside:

```
cp $DUNETPC_DIR/scripts/PandoraSettings_Master_ProtoDUNE.xml MyPandoraSettings_Master_ProtoDUNE.xml
```

```
<pandora>
  <!-- GLOBAL SETTINGS -->
  <IsMonitoringEnabled>true</IsMonitoringEnabled>
  <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
  <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>

  <!-- ALGORITHM SETTINGS -->
  <algorithm type = "LArPreProcessing">
    <algorithm type = "LArVisualMonitoring">
      <algorithm type = "LArMaster">
        <CRSettingsFile>PandoraSettings_Cosmic_Standard.xml</CRSettingsFile>
        <NuSettingsFile>PandoraSettings_TestBeam_ProtoDUNE.xml</NuSettingsFile>
        <SlicingSettingsFile>PandoraSettings_Slicing_ProtoDUNE.xml</SlicingSettingsFile>
        <StitchingTools>
          <tool type = "LArStitchingCosmicRayMerging"><ThreeDStitchingMode>true</ThreeDStitchingMode></tool>
          <tool type = "LArStitchingCosmicRayMerging"><ThreeDStitchingMode>false</ThreeDStitchingMode></tool>
        </StitchingTools>
        <CosmicRayTaggingTools>
          <tool type = "LArCosmicRayTagging"/>
        </CosmicRayTaggingTools>
        <SliceIdTools>
          <tool type = "LArBdtBeamParticleId">
            <BdtName>BeamParticleId</BdtName>
            <BdtFileName>PandoraBdt_v03_12_00.xml</BdtFileName>
            <MinAdaBDTScore>-0.2</MinAdaBDTScore>
          </tool>
        </SliceIdTools>
      </algorithm type = "LArEventValidation">
      <algorithm type = "LArVisualMonitoring">
    </pandora>
```

algorithms

.xml files containing
the big lists of
algorithms running

optional, if monitoring enabled

Note: lines with options and inputs removed here



Visualise events

Aside: You can open one of the others to see the list of algorithms, for example:

```
emacs -nw $DUNETPC_DIR/scripts/PandoraSettings_TestBeam_ProtoDUNE.xml
```

Or your favourite editor!

```
<pandora>
  <!-- GLOBAL SETTINGS -->
  <IsMonitoringEnabled>true</IsMonitoringEnabled>
  <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
  <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>

  <!-- ALGORITHM SETTINGS -->
  <algorithm type = "LArPreProcessing">

    <!-- TwoDReconstruction -->
    <algorithm type = "LArClusteringParent">
      <algorithm type = "LArTrackClusterCreation" description = "ClusterFormation"/>

      <algorithm type = "LArLayerSplitting"/>
      <algorithm type = "LArLongitudinalAssociation"/>
      <algorithm type = "LArTransverseAssociation"/>
      <algorithm type = "LArLongitudinalExtension"/>
      <algorithm type = "LArTransverseExtension"/>
      <algorithm type = "LArCrossGapsAssociation"/>
      <algorithm type = "LArCrossGapsExtension"/>
      <algorithm type = "LArOvershootSplitting"/>
      <algorithm type = "LArBranchSplitting"/>
      <algorithm type = "LArKinkSplitting"/>
      <algorithm type = "LArTrackConsolidation">
        <algorithm type = "LArSimpleClusterCreation" description = "ClusterRebuilding"/>
      etc...
    <!-- VertexAlgorithms -->
    <algorithm type = "LArCandidateVertexCreation">
      <algorithm type = "LArEnergyKickVertexSelection">
        <algorithm type = "LArVertexSplitting">

      <!-- ThreeDTrackAlgorithms -->
      <algorithm type = "LArThreeDTransverseTracks">
        <TrackTools>
          <tool type = "LArClearTracks"/>
          <tool type = "LArLongTracks"/>
          <tool type = "LArOvershootTracks"><SplitMode>true</SplitMode></tool>
          <tool type = "LArUndershootTracks"><SplitMode>true</SplitMode></tool>
          <tool type = "LArOvershootTracks"><SplitMode>false</SplitMode></tool>
          <tool type = "LArUndershootTracks"><SplitMode>false</SplitMode></tool>
          <tool type = "LArMissingTrackSegment"/>
        etc...
      
```

Note: lines with options
and inputs not shown

>100 algorithms+tools
in total in this file



Visualise events

Inside MyPandoraSettings_Master_ProtoDUNE.xml

```
<pandora>
    <!-- GLOBAL SETTINGS -->
    <IsMonitoringEnabled>true</IsMonitoringEnabled>
    <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
    <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>
```

by enabling monitoring

Find these calls instances:

Visual Monitoring algorithms will run

This will visualise the input hits at the beginning

```
<algorithm type = "LArVisualMonitoring">
    <CaloHitListNames>CaloHitListU CaloHitListV CaloHitListW</CaloHitListNames>
    <ShowDetector>true</ShowDetector>
</algorithm>
```

And this will visualise the output PFOs created at the end

```
<algorithm type = "LArVisualMonitoring">
    <ShowCurrentPfos>true</ShowCurrentPfos>
    <ShowDetector>true</ShowDetector>
</algorithm>
```



Visualise events

1) Edit MyPandoraSettings_Master_ProtoDUNE.xml (from previous page)

```
<pandora>
  <!-- GLOBAL SETTINGS -->
  <IsMonitoringEnabled>true</IsMonitoringEnabled>
  <ShouldDisplayAlgorithmInfo>false</ShouldDisplayAlgorithmInfo>
  <SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>
```

2) Tell your .fcl file to use the local version

```
cp $DUNETPC_DIR/job/protoDUNE_reco.fcl  myprotoDUNE_reco.fcl
emacs -nw myprotoDUNE_reco.fcl
and add:
```

```
#Pandora configurations

physics.producers.pandora.HitFinderModuleLabel:           "linecluster"
physics.producers.pandora.ConfigFile:                      "MyPandoraSettings_Master_ProtoDUNE.xml"
```

3) Run your .fcl file again on the events you generated yesterday

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

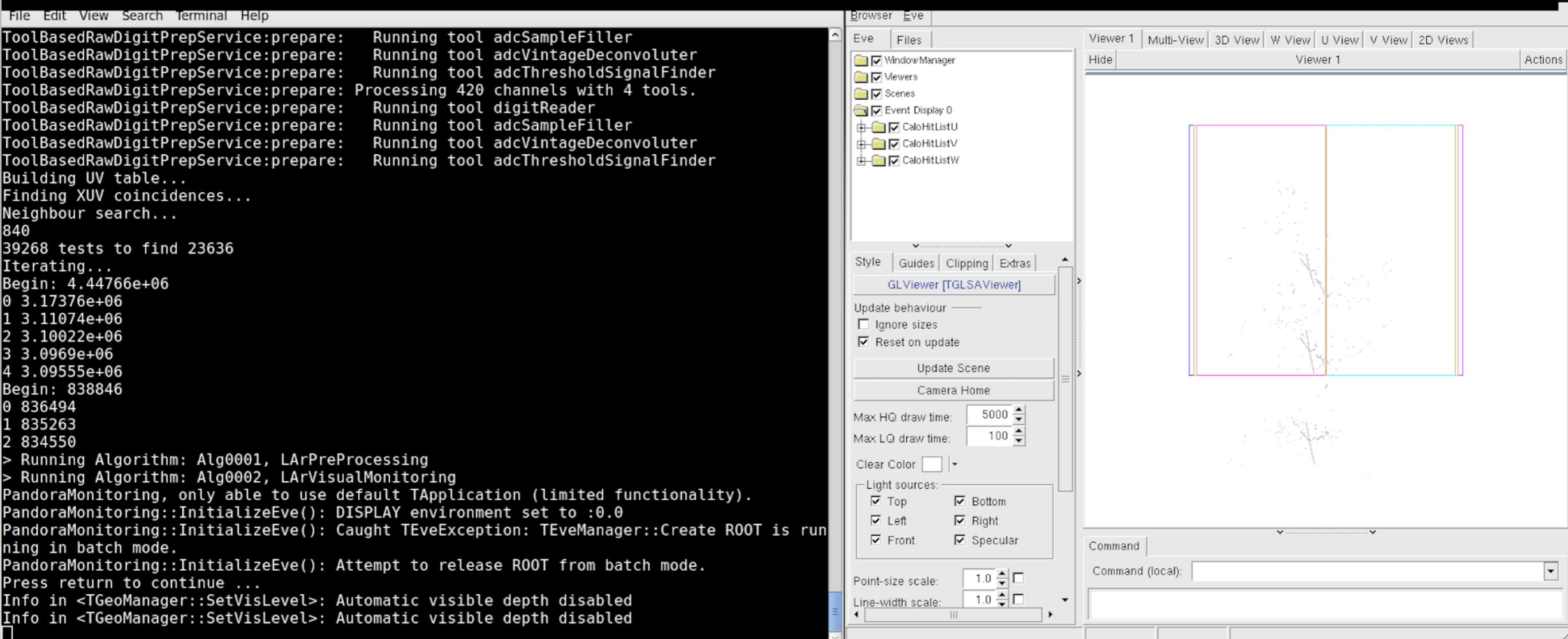


Visualise events

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

This will visualise the input hits

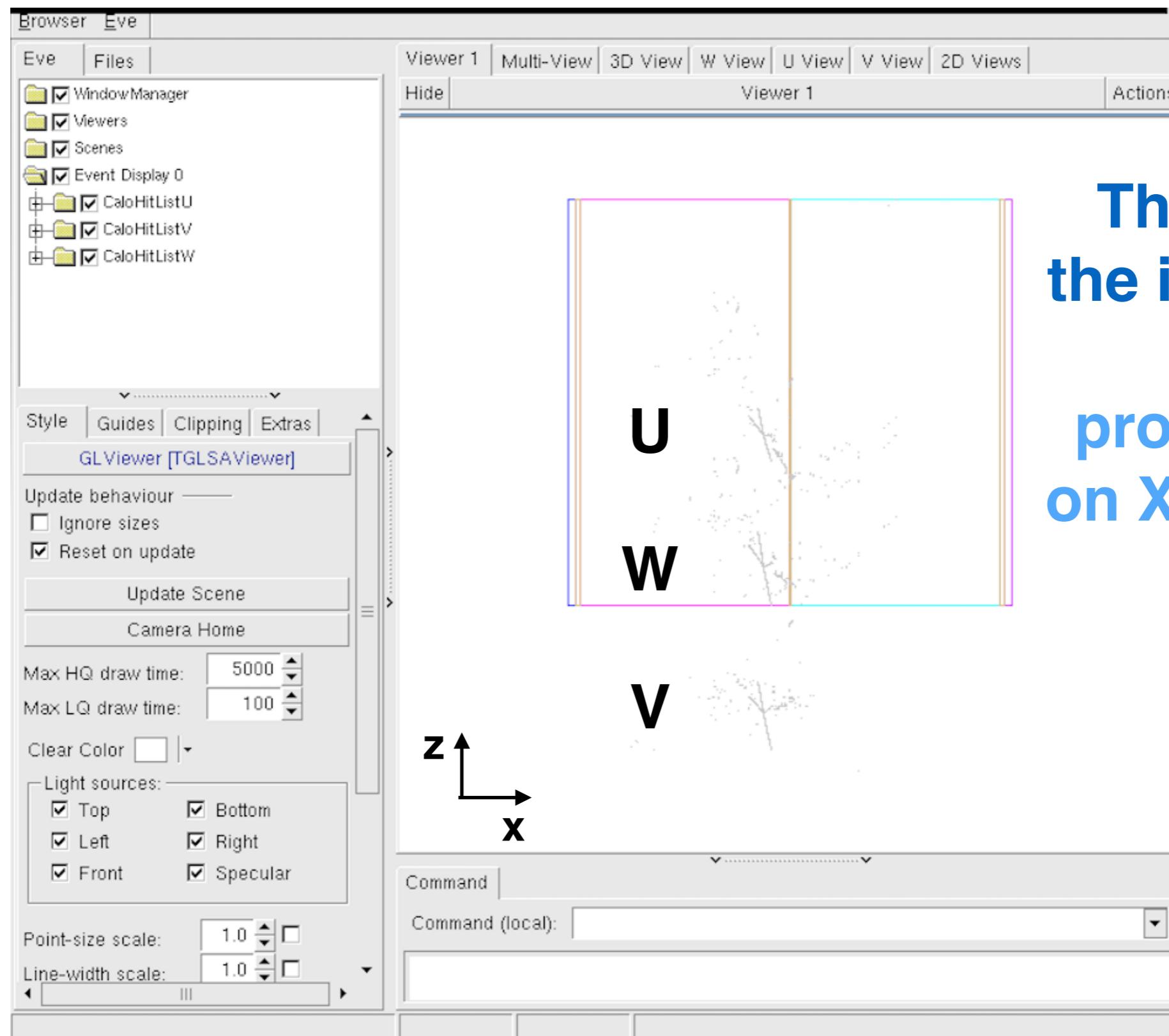
```
<algorithm type = "LArVisualMonitoring">
    <CaloHitListNames>CaloHitListU CaloHitListV CaloHitListW</CaloHitListNames>
    <ShowDetector>true</ShowDetector>
</algorithm>
```



Example: this is an example protoDUNE event of a pion w/o cosmics



Visualise events



These are
the input hits
(U,V
projections
on XW plane)

Example: this is an example protoDUNE event of a pion w/o cosmics

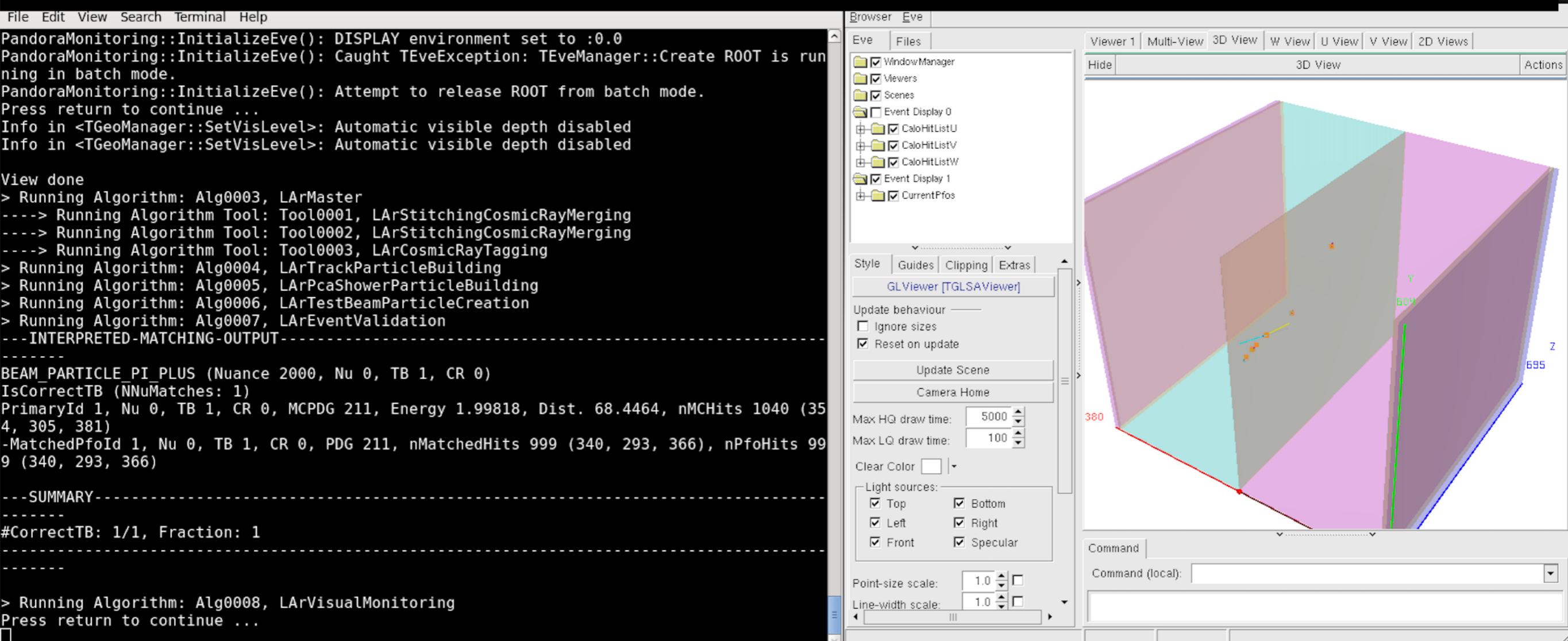


Visualise events

And this will plot the output PFOs created at the end

```
<algorithm type = "LArVisualMonitoring">
    <>ShowCurrentPfos>true</ShowCurrentPfos>
    <ShowDetector>true</ShowDetector>
</algorithm>
```

Click enter!

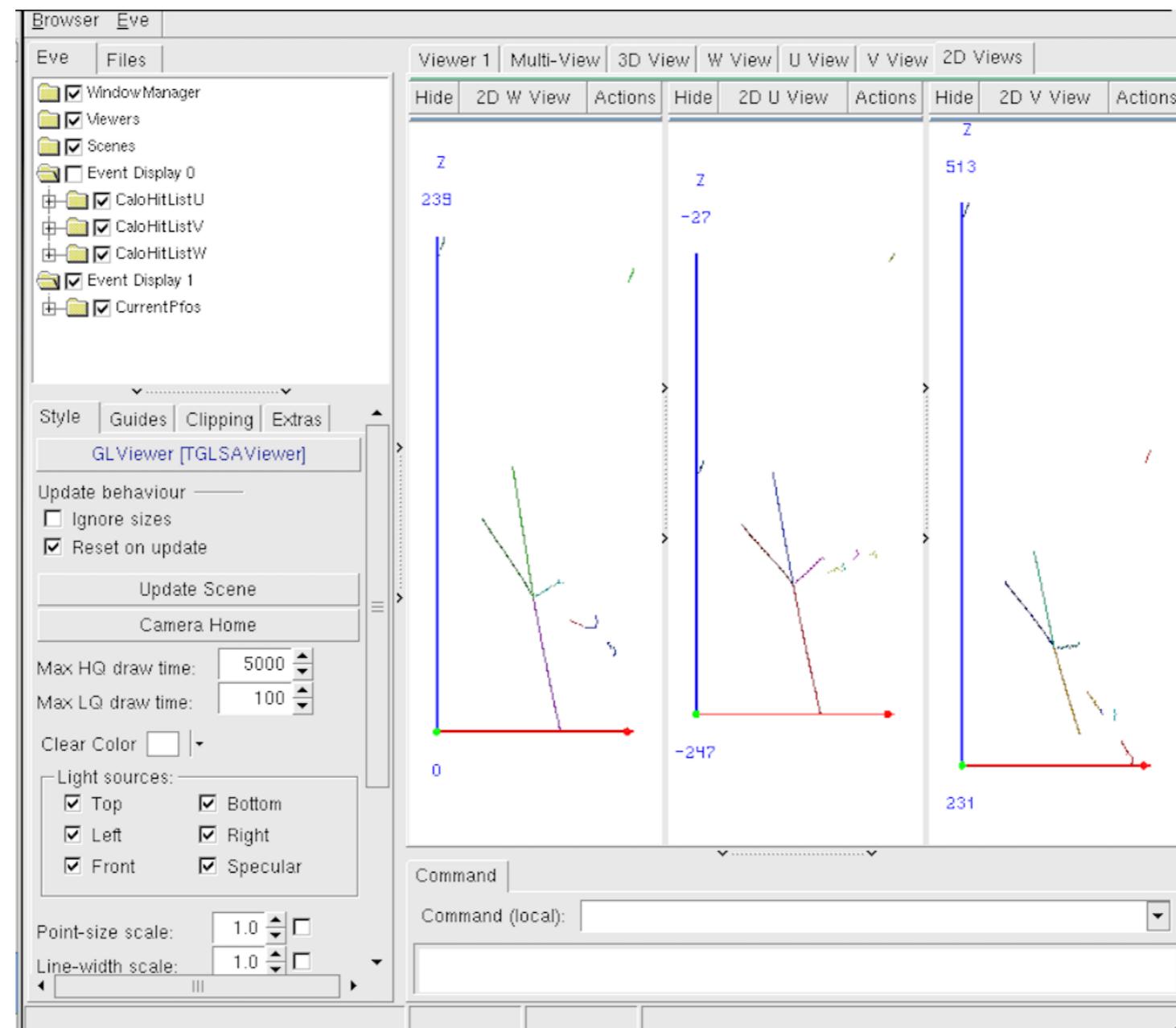


Example: this is an example protoDUNE event of a pion w/o cosmics



Visualise events

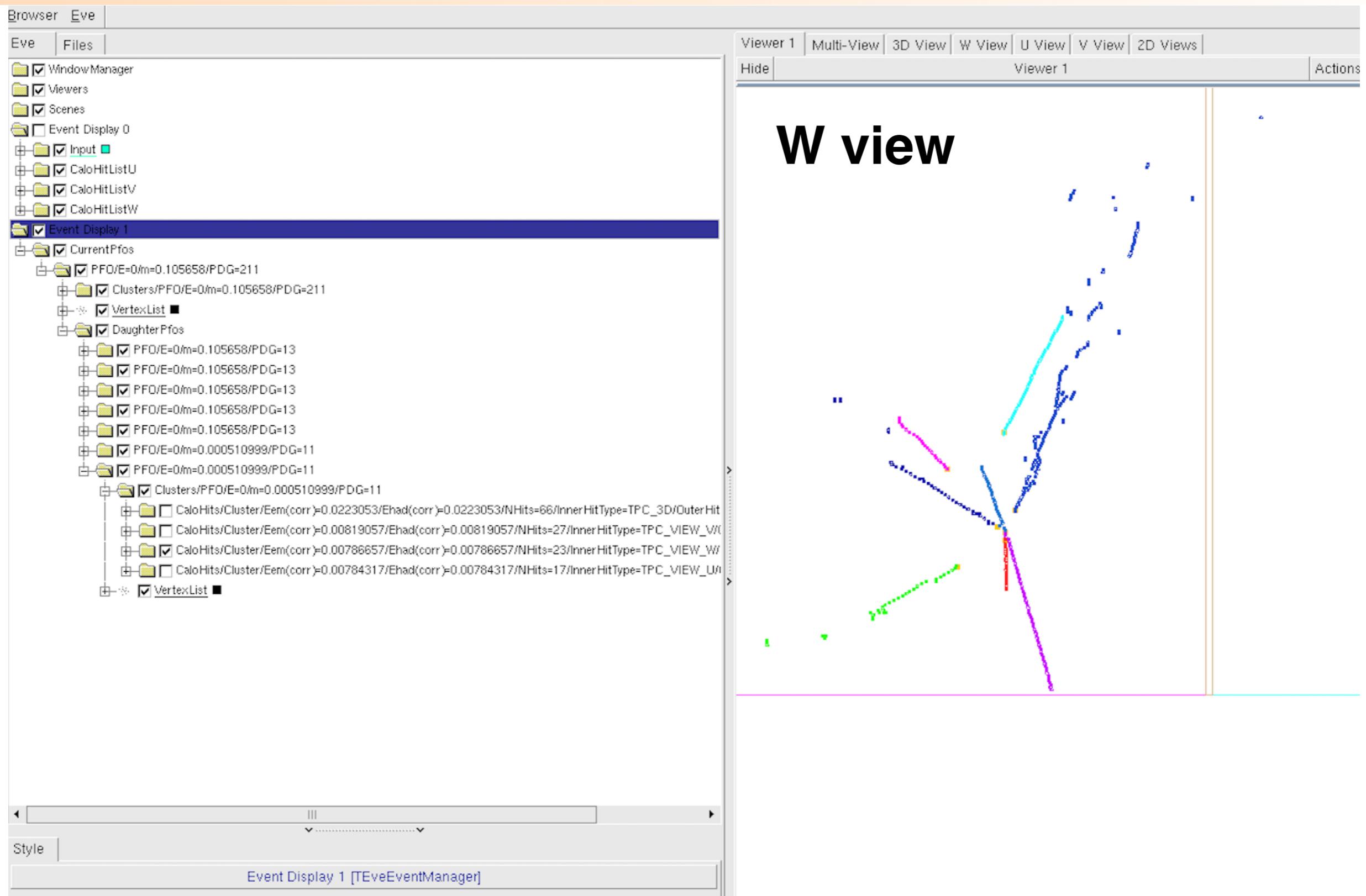
You can visualise the three 2D images separately, W-U-V



Example: this is an example protoDUNE event of a pion w/o cosmics



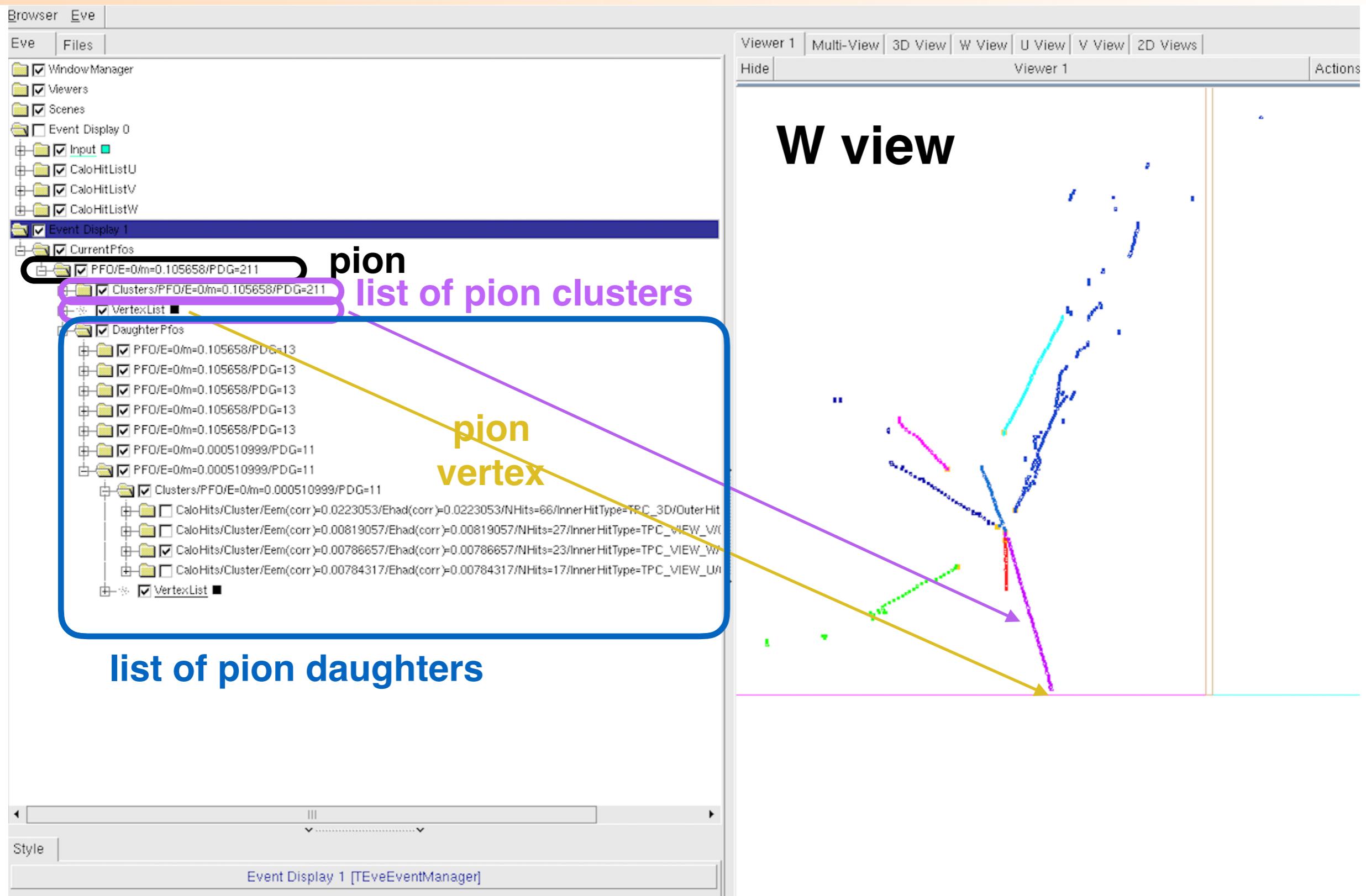
Visualise events



Example: this is an example protoDUNE event of a pion w/o cosmics



Visualise events



Example: this is an example protoDUNE event of a pion w/o cosmics



Visualise events

Browser Eve

Eve | Files

WindowManager
Viewers
Scenes
Event Display 0
Input
CaloHitListU
CaloHitListV
CaloHitListW
Event Display 1
CurrentPfos
PFO/E=0/m=0.105658/PDG=211
Clusters/PFO/E=0/m=0.105658/PDG=211
VertexList
DaughterPfos
PFO/E=0/m=0.105658/PDG=13
PFO/E=0/m=0.105658/PDG=13
PFO/E=0/m=0.105658/PDG=13
PFO/E=0/m=0.105658/PDG=13
PFO/E=0/m=0.105658/PDG=13
PFO/E=0/m=0.000510999/PDG=11
PFO/E=0/m=0.000510999/PDG=11
Clusters/PFO/E=0/m=0.000510999/PDG=11
CaloHits/Cluster/Eem(corr)=0.0223053/Ehad(corr)=0.0223053/NHits=66/InnerHitType=TPC_3D/OuterHit
CaloHits/Cluster/Eem(corr)=0.00819057/Ehad(corr)=0.00819057/NHits=27/InnerHitType=TPC_VIEW_V/
CaloHits/Cluster/Eem(corr)=0.00786657/Ehad(corr)=0.00786657/NHits=23/InnerHitType=TPC_VIEW_W/
CaloHits/Cluster/Eem(corr)=0.00784317/Ehad(corr)=0.00784317/NHits=17/InnerHitType=TPC_VIEW_U/
VertexList

W view

track-like (PDG 13) daughters

shower-like (PDG 11) daughters

each daughter has also a list of clusters and a vertex associated (and its own daughters if applicable)

+ 1 track-like daughter with no W hits

Note: track/shower characterisation is done using information from the three views

Event Display 1 [TEveEventManager]

Example: this is an example protoDUNE event of a pion w/o cosmics



Still time?

**Look in the backup for more exercises
to visualise different stages in events
with cosmics and to print reco-true
matching in Pandora**

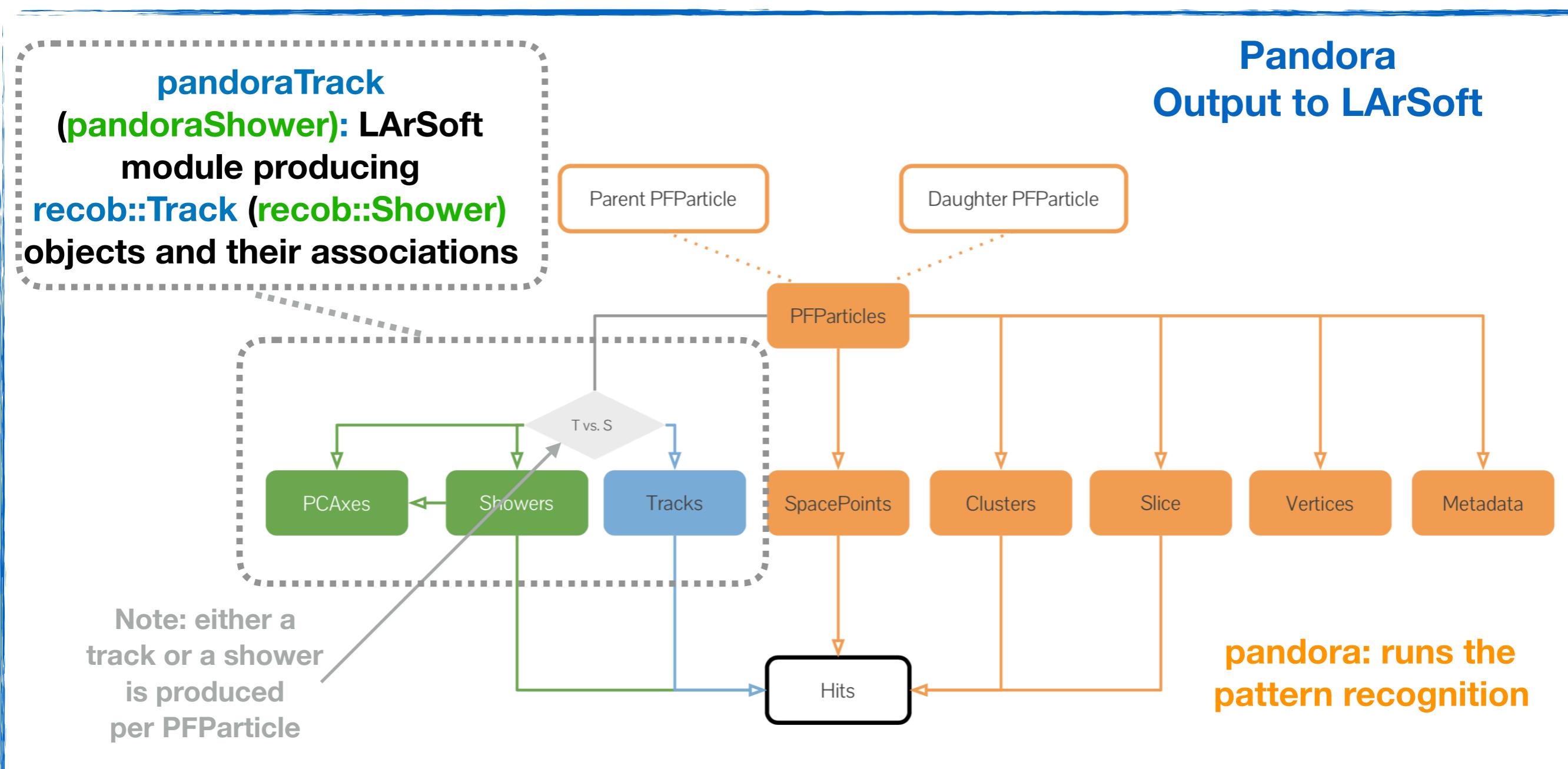


3) Getting familiar with the output



Again the output

This might look a bit overwhelming at first...





Event Dump

But it isn't that bad once you get used to it! So lets dump it!

This is an example of what you can see the reco file contains using eventdump.fcl

PROCESS NAME	MODULE_LABEL..	PRODUCT INSTANCE NAME..	DATA PRODUCT TYPE.....	.SIZE
Reco.....	pandora.....	std::vector<recob::PFParticle>.....	...12
Reco.....	pandora.....	std::vector<recob::Vertex>.....	...11
Reco.....	pandora.....	std::vector<larpandoraobj::PFParticleMetadata>.....	...12
Reco.....	pandora.....	std::vector<recob::SpacePoint>.....	.508
Reco.....	pandora.....	std::vector<recob::Cluster>.....	...3
Reco.....	pandora.....	std::vector<anab::T0>.....	...0
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::SpacePoint, void>.....	.508
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::Vertex, void>.....	...11
Reco.....	pandora.....	art::Assns<recob::Cluster, recob::Hit, void>.....	.620
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::Cluster, void>.....	...33
Reco.....	pandora.....	art::Assns<recob::PFParticle, larpandoraobj::PFParticleMetadata, void>.....	...12
Reco.....	pandora.....	art::Assns<recob::PFParticle, anab::T0, void>.....	...0
Reco.....	pandora.....	art::Assns<recob::SpacePoint, recob::Hit, void>.....	.508
Reco.....	pandoraTrack..	std::vector<recob::Track>.....	...7
Reco.....	pandoraTrack..	art::Assns<recob::PFParticle, recob::Track, void>.....	...7
Reco.....	pandoraTrack..	art::Assns<recob::Track, recob::Hit, void>.....	.345
Reco.....	pandoraTrack..	art::Assns<recob::Track, recob::Hit, recob::TrackHitMeta>.....	.345
Reco.....	pandoraShower.	std::vector<recob::Shower>.....	...4
Reco.....	pandoraShower.	std::vector<recob::PCAxis>.....	...4
Reco.....	pandoraShower.	art::Assns<recob::Shower, recob::Hit, void>.....	.257
Reco.....	pandoraShower.	art::Assns<recob::PFParticle, recob::PCAxis, void>.....	...4
Reco.....	pandoraShower.	art::Assns<recob::PFParticle, recob::Shower, void>.....	...4
Reco.....	pandoraShower.	art::Assns<recob::Shower, recob::PCAxis, void>.....	...4

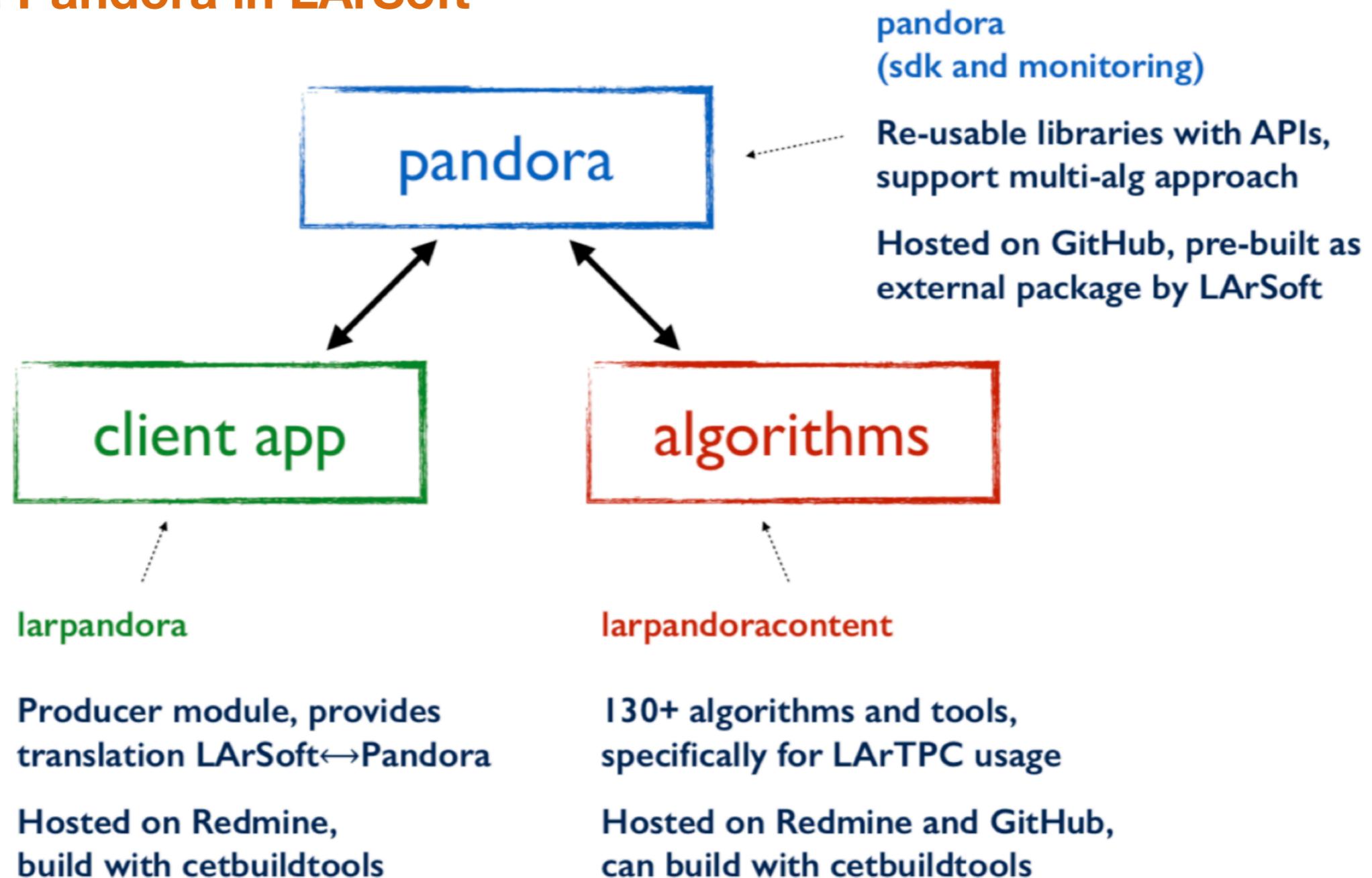
And now we are going to use a Pandora Event Dump that translates these objects into hierarchies, written by AndySmith (Cambridge student)



Pandora Event Dump

And now we are going to use a Pandora Event Dump that translates these objects into hierarchies, written by AndySmith (Cambridge student) and living in *larpandora*

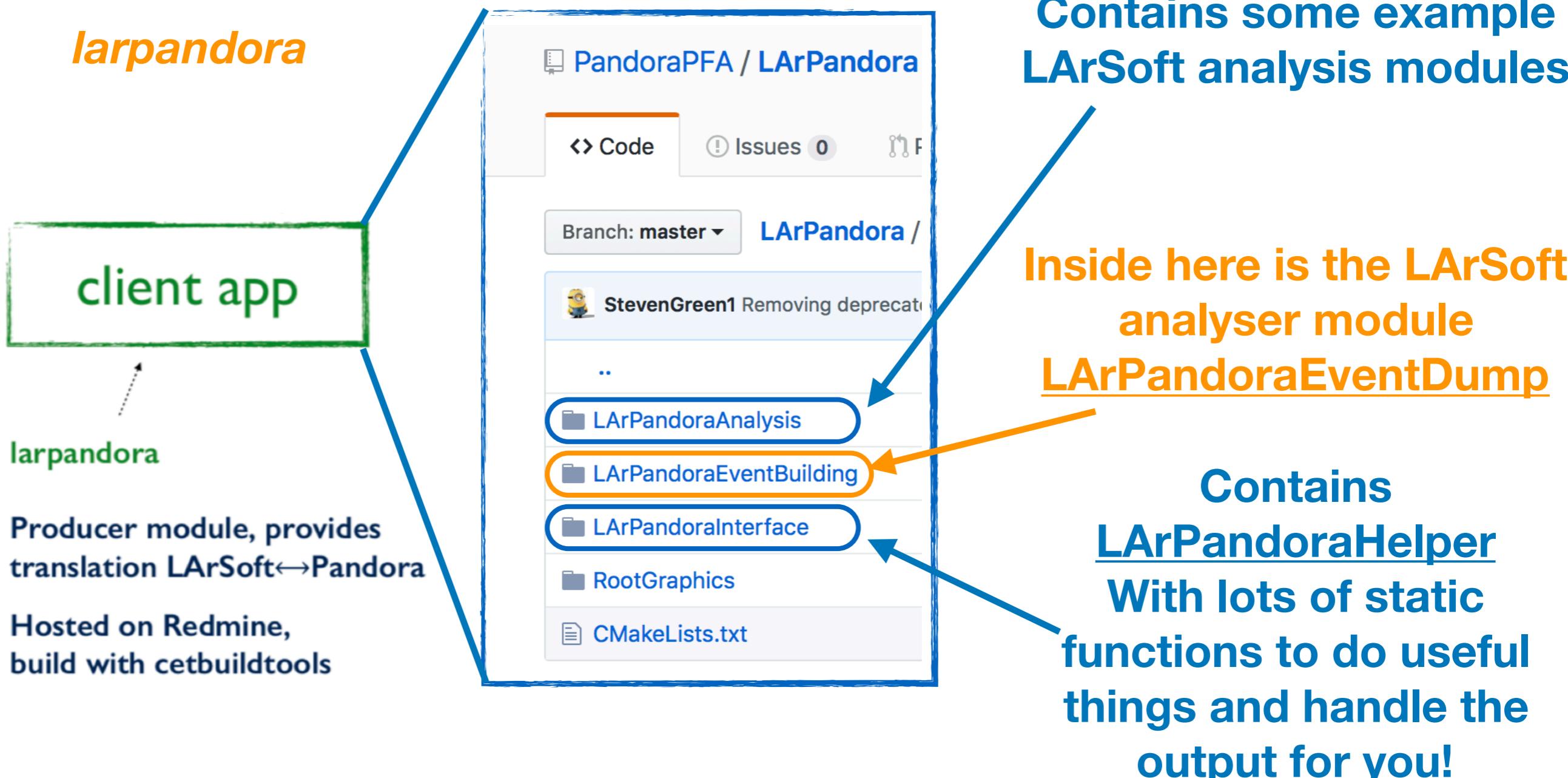
Reminder: Pandora in LArSoft





Pandora Event Dump

And now we are going to use a Pandora Event Dump that translates these objects into hierarchies, written by AndySmith (Cambridge student) and living in *larpandora*



<https://github.com/PandoraPFA/LArPandora/tree/master/larpandora>



Pandora Event Dump

**Write a simple .fcl file
to run the LArSoft
analyser module
LArPandoraEventDump**

For example: write this in `run_pandora_event_dump.fcl`

And run it!
`lar -c
run_pandora_event_
dump.fcl
my_reco_events.root`

```
BEGIN_PROLOG
pandora_event_dump:
{
    module_type: "LArPandoraEventDump" Module name
}

dump: @local::pandora_event_dump
dump.PandoraLabel: "pandora" Pattern recognition producer name
dump.TrackLabel: "pandoraTrack" recob::Tracks producer name
dump.ShowerLabel: "pandoraShower" recob::Showers producer name
dump.VerboseLevel: "detailed"
END_PROLOG

#include "services_dune.fcl"
services:
{
    scheduler: { defaultExceptions: false }
    RandomNumberGenerator: {} #ART native random number generator
    FileCatalogMetadata: @local::art_file_catalog_mc
}

process_name: LArPandoraEventDump Module name
source:
{
    module_type: RootInput
}
physics:
{
    analyzers:
    {
        dump: @local::dump
    }
    stream1: [ dump ]
    end_paths: [ stream1 ]
}
outputs: {}
```



Pandora Event Dump

The output will be something like this:

**First: summary of the reconstruction products
(similar to what you see with eventdump.fcl)**

```
run: 1 subRun: 0 event: 1
pandora
-----
N PFParticles : 261
N SpacePoints : 42761
N Clusters   : 664
N Vertices    : 261
N Metadata    : 261
N Tracks      : 110
N Showers     : 151
N PCAxes      : 151
```

PFParticle

- Key	0	ID of the particle
- Id	0	Is a primary particle (no parent)
- Primary		
- PDG	13	This indicates is track-like
<hr/>		
- # SpacePoint	3791	Number of associated SpacePoints (3D hits)
- # Cluster	9	Number of associated Clusters (>3 means crosses different “TPCs”)
- # Vertex	1	
- # Track	1	Since is track-like, only a recob::Track is created
- # Shower	0	
- # PCAxis	0	
- # Metadata	1	
-- Property	SliceIndex, value 1	Metadata info
-- Property	TestBeamScore, value -0.973129	
<hr/>		
- # Daughters	11	Number of daughters
<hr/>		

Then information about each PFParticle



Pandora Event Dump

```
-----  
PFParticle  
-----  
- Key 0  
- Id 0  
- Primary  
- PDG 13  
-----  
- # SpacePoint 3791  
- # Cluster 9  
- # Vertex 1  
- # Track 1  
- # Shower 0  
- # PCAxis 0  
- # Metadata 1  
-- Property SliceIndex, value 1  
-- Property TestBeamScore, value -0.973129  
-----  
- # Daughters 11  
-----
```

```
-----  
PFParticle  
-----  
- Key 87  
- Id 87  
- Parent 0  
- PDG 11  
-----  
- # SpacePoint 17  
- # Cluster 3  
- # Vertex 1  
- # Track 0  
- # Shower 1  
- # PCAxis 1  
- # Metadata 1  
-----  
- # Daughters 0  
-----
```

The hierarchy...

Parent

And for each daughter...

etc...



Pandora Event Dump

PFParticle

```
- Key 1  
- Id 1  
- Primary  
- PDG 13  
  
- # SpacePoint 3080  
- # Cluster 13  
- # Vertex 1  
- # Track 1  
- # Shower 0  
- # PCAxis 0  
- # Metadata 1  
--- Property IsClearCosmic, value 1  
  
- # Daughters 9
```

Clear Cosmic

And the metadata

PFParticle

```
- Key 43  
- Id 43  
- Primary  
- PDG 211  
  
- # SpacePoint 132  
- # Cluster 3  
- # Vertex 1  
- # Track 1  
- # Shower 0  
- # PCAxis 0  
- # Metadata 1  
--- Property IsTestBeam, value 1  
--- Property SliceIndex, value 4  
--- Property TestBeamScore, value 0.0104636  
  
- # Daughters 6
```

Test Beam Particle

PFParticle

```
- Key 0  
- Id 0  
- Primary  
- PDG 13  
  
- # SpacePoint 3791  
- # Cluster 9  
- # Vertex 1  
- # Track 1  
- # Shower 0  
- # PCAxis 0  
- # Metadata 1  
--- Property SliceIndex, value 1  
--- Property TestBeamScore, value -0.97312
```

No Clear Cosmic & No Test Beam = Ambiguous Cosmic



Pandora Event Dump

Exercise: run the Pandora Event Dump on the events you have visualised before and try to match the information from both sources

The Pandora Event Dump has other options, you can be as verbose as you want in the output printed out! Have a look at the functions and options inside it [here!](#)



More tools and analyser examples



More Tools

There are lots of functions already written in [LArPandoraHelper](#) that can be useful

For example, have a look at [LArPandoraHelper::BuildPFParticleHitMaps](#)
to see how to fold in daughters when collecting PFParticle maps
(which you need to do in your calorimetry analyses!)

Lots of things are already done for you! Have a look when writing your analysis
to see if there are things you can use already

```
/**  
 * @brief Build mapping between PFParticles and Hits using PFParticle/SpacePoint/Hit maps  
 *  
 * @param particleVector the input vector of PFParticle objects  
 * @param particlesToSpacePoints the input map from PFParticle to SpacePoint objects  
 * @param spacePointsToHits the input map from SpacePoint to Hit objects  
 * @param particlesToHits the output map from PFParticle to Hit objects  
 * @param hitsToParticles the output map from Hit to PFParticle objects  
 * @param daughterMode treatment of daughter particles in construction of maps  
 */  
static void BuildPFParticleHitMaps(const PFParticleVector &particleVector, const PFParticlesToSpacePoints &particlesToSpacePoints,  
    const SpacePointsToHits &spacePointsToHits, PFParticlesToHits &particlesToHits, HitsToPFParticles &hitsToParticles,  
    const DaughterMode daughterMode = kUseDaughters);
```



And Analyzer examples

See also some examples of using the output and writing an analyzer from scratch

1. From Paraguay workshop in 2018 [here](#)
2. From the 2018 LArSoft Manchester workshop (by Leigh) [here](#)
3. From the Pandora Workshop in 2016 (so a bit outdated) [here](#)

A Usage Example

[Talk by Leigh here](#)

- There is a skeleton module you can use to get started
 - `dunetcpc/dune/Protodune/Analysis/BeamExample/BeamExample_module.cc`

```
bool beamTriggerEvent = false;
// If this event is MC then we can check what the true beam particle is
if(!evt.isRealData()){
    // Get the truth utility to help us out
    protoana::ProtoDUNETruthUtils truthUtil;
    // Firstly we need to get the list of MCTruth objects from the generator. The standard protoDUNE
    // simulation has fGeneratorTag = "generator"
    auto mcTruths = evt.getValidHandle<std::vector<simb::MCTruth>>(fGeneratorTag);
    // mcTruths is basically a pointer to an std::vector of simb::MCTruth objects. There should only be one
    // of these, so we pass the first element into the function to get the good particle
    const simb::MCParticle* geantGoodParticle = truthUtil.GetGeantGoodParticle(*mcTruths)[0],evt);
    if(geantGoodParticle != 0x0){
        std::cout << "Found GEANT particle corresponding to the good particle with pdg = " << geantGoodParticle->PdgCode() << std::endl;
    }
}
else{
    // For data we can see if this event comes from a beam trigger
    beamTriggerEvent = dataUtil.IsBeamTrigger(evt);
    if(beamTriggerEvent){
        std::cout << "This data event has a beam trigger" << std::endl;
    }
}
```

For MC, it finds the triggered true particle using the truth utility

For Data, it looks for a beam trigger using the data utility





Lost in the dark?

Contact us!

General:

pandora@hep.phy.cam.ac.uk

Pandora SDK Development

John Marshall (John.Marshall@warwick.ac.uk)
Mark Thomson (thomson@hep.phy.cam.ac.uk)

LAr TPC algorithm development

John Marshall (John.Marshall@warwick.ac.uk)
Andy Blake (a.blake@lancaster.ac.uk)

DUNE FD Integration

Lorena Escudero (escudero@hep.phy.cam.ac.uk)

ProtoDUNE Integration

Steven Green (sg568@hep.phy.cam.ac.uk)

MicroBooNE Integration

Andy Smith (asmith@hep.phy.cam.ac.uk)

Other team members

MicroBooNE: Joris Jan de Vries, Jack Anthony
ProtoDUNE: Stefano Vergani



<https://github.com/PandoraPFA>



<https://pandorapfa.slack.com>



UNIVERSITY OF
CAMBRIDGE

Lancaster
University



WARWICK
THE UNIVERSITY OF WARWICK



Backup



Simulate events

You can do something like this to generate your own events

```
cd MyTestAnalysis

source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh
setup larsoft v07_08_00 -q e17:prof
setup dunetpc v07_08_00 -q e17:prof
mrb newDev
source localProducts_larsoft_v07_08_00_e17_prof/setup
```

i) set up work environment

```
lar -c mcc11_gen_protoDune_beam_cosmics_p5GeV.fcl -n 10
[ -o my_generated_events.root] #optional

lar -c protoDUNE_g4.fcl -n 10 my_generated_events.root
[ -o my_generated_events_g4.root ] #optional

lar -c protoDUNE_detsim.fcl -n 10 my_generated_events_g4.root
[ -o my_generated_events_g4_detsim.root ] #optional
```

ii) generate events



I) Run the reconstruction

What is happening...

```
lar -c protoDUNE_reco.fcl -n 10 my_generated_events.root  
[-o my_reconstructed_events.root]
```

Initialization messages

```
DuneDeconvolutionService::ctor: DuneDeconvolutionService:  
DuneDeconvolutionService::ctor: LogLevel: 1  
DuneRoiBuildingService::ctor: DuneRoiBuildingService:  
DuneRoiBuildingService::ctor: LogLevel: 1  
DuneRoiBuildingService::ctor: NSigmaStart: 3  
DuneRoiBuildingService::ctor: NSigmaEnd: 1  
DuneRoiBuildingService::ctor: PadLow: 50  
DuneRoiBuildingService::ctor: PadHigh: 50  
StandardAdcWireBuildingService::ctor: StandardAdcWireBuildingService:  
StandardAdcWireBuildingService::ctor: LogLevel: 1  
Info in <TGeoManager::Import>: Reading geometry from file: /cvmfs/dune.opensciencegrid.org/products/dune/  
dunetpc/v07_07_00_01/gdml/protodune_v5.gdml  
Info in <TGeoManager::TGeoManager>: Geometry GDMLImport, Geometry imported from GDML created  
Info in <TGeoManager::SetTopVolume>: Top volume is volWorld. Master volume is volWorld  
Info in <TGeoNavigator::BuildCache>: --- Maximum geometry depth set to 100  
Info in <TGeoManager::CheckGeometry>: Fixing runtime shapes...  
Info in <TGeoManager::CheckGeometry>: ...Nothing to fix  
Info in <TGeoManager::CloseGeometry>: Counting nodes...  
Info in <TGeoManager::Voxelize>: Voxelizing...  
Info in <TGeoManager::CloseGeometry>: Building cache...  
Info in <TGeoManager::CountLevels>: max level = 5, max placements = 1148  
Info in <TGeoManager::CloseGeometry>: 36144 nodes/ 5401 volume UID's in Geometry imported from GDML  
Info in <TGeoManager::CloseGeometry>: -----modeler ready-----  
GeoApaChannelGroupService::ctor: Group 0 (apa0) has 2080 channels from 3/4 readout planes.  
GeoApaChannelGroupService::ctor: Group 1 (apa1) has 2080 channels from 3/4 readout planes.  
GeoApaChannelGroupService::ctor: Group 2 (apa2) has 2080 channels from 3/4 readout planes.  
GeoApaChannelGroupService::ctor: Group 3 (apa3) has 2080 channels from 3/4 readout planes.  
GeoApaChannelGroupService::ctor: Group 4 (apa4) has 2080 channels from 3/4 readout planes.  
GeoApaChannelGroupService::ctor: Group 5 (apa5) has 2080 channels from 3/4 readout planes.
```

General services

Geometry

etc...



I) Run the reconstruction

What is happening...

```
lar -c protoDUNE_reco.fcl -n 10 my_generated_events.root  
[-o my_reconstructed_events.root]
```

Opening input file...

```
28-Oct-2018 18:23:12 GMT Initiating request to open input file "protoDUNE_pion_2GeV_mono_G4_detsim_events.root"  
28-Oct-2018 18:23:12 GMT Initiating request to open input file "protoDUNE_pion_2GeV_mono_G4_detsim_events.root"  
28-Oct-2018 18:23:13 GMT Opened input file "protoDUNE_pion_2GeV_mono_G4_detsim_events.root"  
28-Oct-2018 18:23:13 GMT Opened input file "protoDUNE_pion_2GeV_mono_G4_detsim_events.root"
```

Processing the events.

Some producers will be printing out messages, for example:

```
%MSG-e DataProviderAlg: PMAlgTrackMaker:pmtrack@BeginModule 28-Oct-2018 18:23:35 GMT run: 1 subRun: 0 event: 1  
Wires data not set in the cryo:0 tpc:0 plane:2  
%MSG
```

Closing input and output files...

```
28-Oct-2018 18:26:10 GMT Closed output file "protoDUNE_pion_2GeV_mono_G4_detsim_events_reco.root"  
28-Oct-2018 18:26:10 GMT Closed output file "protoDUNE_pion_2GeV_mono_G4_detsim_events_reco.root"  
28-Oct-2018 18:26:10 GMT Closed input file "protoDUNE_pion_2GeV_mono_G4_detsim_events.root"  
28-Oct-2018 18:26:10 GMT Closed input file "protoDUNE_pion_2GeV_mono_G4_detsim_events.root"  
  
DataPrepModule::endJob: # events processed: 10  
DataPrepModule::endJob: # events skipped: 0
```



I) Run the reconstruction

What is happening...

```
lar -c protoDUNE_reco.fcl -n 10 my_generated_events.root  
[-o my_reconstructed_events.root]
```

Summary print out from the art::TimeTracker service

TimeTracker printout (sec)	Min	Avg	Max	Median	RMS	nEvts
Full event	10.9239	16.4927	30.7107	13.8305	6.03821	10
source:RootInput(read)	0.000281449	0.000515754	0.00146504	0.000427652	0.000323607	10
reco:rns:RandomNumberSaver	2.6142e-05	5.42986e-05	0.000274468	3.07545e-05	7.3424e-05	10
reco:ophit:OpHitFinder	0.000472357	0.0176818	0.168036	0.00110279	0.0501185	10
reco:opflash:OpFlashFinder	0.000192166	0.000496848	0.0023595	0.000310921	0.000622998	10
reco:caldata:DataPrepModule	8.44578	8.69662	10.8608	8.45596	0.721432	10
reco:gaushit:GausHitFinder	0.28158	0.480787	0.828892	0.455886	0.146557	10
reco:reco3d:SpacePointSolver	0.0127969	2.40228	17.5862	0.0844079	5.32705	10
reco:hitpdune:DisambigFromSpacePoints	0.011602	0.0208112	0.0369639	0.0189795	0.0081143	10
reco:linecluster:LineCluster	0.0125113	0.0201407	0.0328741	0.0186305	0.00655096	10
reco:emtrkmichelid:EmTrackMichelId	1.3143	2.51923	3.50061	2.6984	0.775743	10
reco:pandora:StandardPandora	0.179093	0.411216	0.670182	0.408648	0.146957	10
reco:pandoraTrack:LArPandoraTrackCreation	0.00202527	0.00458205	0.00726229	0.00490643	0.00181189	10
reco:pandoraShower:LArPandoraShowerCreation	0.000309023	0.000727246	0.0019323	0.000435659	0.000519239	10
reco:pandoracalo:Calorimetry	0.00116318	0.00648228	0.0164835	0.0042571	0.0052587	10
reco:pandorapid:Chi2ParticleID	0.000159165	0.00107007	0.00864384	0.00023664	0.00252532	10
reco:pandoracali:CalibrationEdXPDSP	0.00050625	0.651673	6.5022	0.00184645	1.95018	10
reco:pandoracalipid:Chi2ParticleID	0.000110441	0.000182945	0.000264772	0.000183379	5.83571e-05	10
reco:pmtrack:PMAlgTrackMaker	0.149092	1.18471	4.11319	0.815881	1.18543	10
reco:pmtrackcalo:Calorimetry	6.4012e-05	0.00387695	0.0141003	0.00107946	0.00518883	10
reco:pmtrackpid:Chi2ParticleID	5.6334e-05	0.000168095	0.000272963	0.000165131	5.80321e-05	10
reco:pmtrackcali:CalibrationEdXPDSP	4.3335e-05	0.00101092	0.00232963	0.000639647	0.000922123	10
reco:pmtrackcalipid:Chi2ParticleID	3.0099e-05	0.000114284	0.000223515	0.000114486	5.80596e-05	10
reco:TriggerResults:TriggerResultInserter	1.079e-05	1.18981e-05	1.8133e-05	1.12365e-05	2.09407e-06	10
end_path:out1:RootOutput	1.276e-06	1.5665e-06	2.907e-06	1.43e-06	4.54485e-07	10
end_path:out1:RootOutput(write)	0.0282791	0.0680769	0.114847	0.0688115	0.0321029	10

Note: We will come back to this slide once we have looked inside the .fcl file



I) Run the reconstruction

What is happening...

```
lar -c protoDUNE_reco.fcl -n 10 my_generated_events.root  
[-o my_reconstructed_events.root]
```

Summary print out from the art::MemoryTracker service

```
=====
```

```
MemoryTracker summary (base-10 MB units used)
```

```
Peak virtual memory usage (VmPeak) : 3062.08 MB  
Peak resident set size usage (VmHWM): 1573.35 MB
```

```
=====
```

And give a summary of the success of the process

```
TrigReport ----- Event Summary -----  
TrigReport Events total = 10 passed = 10 failed = 0
```

```
TrigReport ----- Modules in End-Path: end_path -----  
TrigReport Trig Bit# Run Success Error Name  
TrigReport 0 0 10 10 0 out1
```

```
TimeReport ----- Time Summary ---[sec]---  
TimeReport CPU = 222.723141 Real = 175.510349
```

```
MemReport ----- Memory Summary ---[base-10 MB]----  
MemReport VmPeak = 3062.08 VmHWM = 1573.35
```

Art has completed and will exit with status 0.

0 is the number
you will always want to see!



2) Let's look at the reco .fcl file

```
cp $DUNETPC_DIR/job/protoDUNE_reco.fcl myprotoDUNE_reco.fcl
```

And open it!

```
#include ...  
  
process_name: Reco  
  
services:  
{  
    ...  
}  
  
physics:  
{  
    producers:  
    {  
        ...  
    }  
  
    reco: [...] Define the producer modules we are going to run...  
  
    #define the output stream, there could be more than one if using filters  
    stream1: [ out1 ]  
  
    #trigger_paths is a keyword and contains the paths that modify the art::Event, i.e. filters and producers  
    trigger_paths: [reco]  
  
    #end_paths is a keyword and contains the paths that do not modify the art::Event, i.e. analysers and output  
    end_paths: [stream1]  
}  
  
outputs:  
{  
    out1:  
    {  
        ...  
    }  
    ...  
}
```

Art services

Define the producer modules we are going to run...

...which are specified and ordered in this list

Output file name

etc.

General structure



2) Let's look at the reco .fcl file

```
emacs -nw myprotoDUNE_reco.fcl
```

```
#include "services_dune.fcl"
#include "caldata_dune.fcl"
#include "hitfindermodules_dune.fcl"
#include "SpacePointSolver_dune.fcl"
#include "cluster_dune.fcl"
#include "trackfindermodules_dune.fcl"
#include "pandoramodules_dune.fcl" ← specific Pandora information is in another .fcl file (we will see it later)
#include "calorimetry_dune10kt.fcl"
#include "calibration_dune.fcl"
#include "featurelabelingmodules.fcl"
#include "particleid.fcl"
#include "mctrutht0matching.fcl"
#include "t0reco.fcl"
#include "opticaldetectormodules_dune.fcl"
#include "showerfindermodules_dune.fcl"
#include "emshower3d.fcl"

#include "protodune_tools_dune.fcl"

process_name: Reco ← process name (this is the step we are adding in the production chain)

services:
{
    # Load the service that manages root files for histograms.
    TFileService: { fileName: "reco_protoDUNE_hist.root" }
    TimeTracker: {} ← Previous print out
    MemoryTracker: {}

    RandomNumberGenerator: {} #ART native random number generator
    message: @local::dune_message_services_prod_debug
    FileCatalogMetadata: @local::art_file_catalog_mc
                            @table::protodune_reco_services
}
```

specific Pandora information is in another .fcl file
(we will see it later)

process name
(this is the step we are adding in the production chain)

general services setup



2) Let's look at the reco .fcl file

```
producers:  
{  
# random number saver  
rns: { module_type: RandomNumberSaver }  
# convert raw::RawDigit to recob::wire  
caldata: @local::producer_adcpref  
# actual hit finder  
gaushit: @local::protodunespmc_gaushitfinder  
fasthit: @local::dunefd_fasthitfinder  
# space point solver  
reco3d: @local::protodunespmc_spacepointsolver  
# actual disambiguation  
hitfd: @local::dunefd_hitfinderfd  
hitpdune: @local::pdune_disambigfromsp  
# 3d dbscan  
dbcluster: @local::protodunespmc_dbcluster3d  
# event feature labeling  
emtrkmichelid: @local::protodune_emtrkmichelid  
# reconstruction using disambiguated hits  
linecluster: @local::dune35t_linecluster  
calo: @local::dune35t_calomc  
pandora: @local::protodune_pandora  
pandoraTrack: @local::dune_pandoraTrackCreation  
pandoraShower: @local::dune_pandoraShowerCreation  
pandoracalo: @local::dune10kt_calomc  
pandorapid: @local::standard_chi2pid  
pandoracali: @local::protodunespmc_calibrationdedx  
pandoracalipid: @local::standard_chi2pid  
pmtrack: @local::dunefd_pmalgtrackmaker  
pmtrackcalo: @local::dune10kt_calomc  
pmtrackpid: @local::standard_chi2pid  
pmtrackcali: @local::protodunespmc_calibrationdedx  
pmtrackcalipid: @local::standard_chi2pid  
pmtrajfit: @local::dunefd_pmalgtrajfitter  
pmtrajfitcalo: @local::dune10kt_calomc  
pmtrajfitpid: @local::standard_chi2pid  
# photon detector reconstruction  
ophit: @local::protodune_ophit  
opflash: @local::protodune_opflash  
}
```

```
emacs -nw myprotoDUNE_reco.fcl
```

Producers: the different modules we can call in the process to perform the different tasks

They are referenced here as **@local::something** and this definition can be found in other .fcl files e.g. we will see later **@local::protodune_pandora**



2) Let's look at the reco .fcl file

And these are the producers that actually run in our process, called “reco” (so the ones from the list in previous page that we are including in this sequence are the ones actually running)

```
emacs -nw myprotoDUNE_reco.fcl
```

```
reco: [ rns,
    #optical hits and flashes
    ophit, opflash,
    #TPC wire signals
    caldata,
    #hit reconstruction
    gaushit, #fasthit,
    #space point solver
    reco3d,
    #real disambiguation
    hitpdune,
    # 3d dbcluster
    #dbcluster,
    #cluster reco
    linecluster, →
    #feature labeling
    emtrkmichelid,
    #pandora
    pandora, pandoraTrack, pandoraShower,
    pandoracalo, pandorapid, pandoracali, pandoracalipid,
    #pmattrack
    pmtrack, pmtrackcalo, pmtrackpid, pmtrackcali, pmtrackcalipid
    #pmtrajfit, pmtrajfitcalo, pmtrajfitpid
    #shower reconstruction
    #blurredcluster, emshower, emshower3d, mergeemshower3d
]
```

Hit finder producers (hit reconstruction).
Output: collection of hits

Cluster finder producer.
Several outputs: clusters and associations,
but also collection of recob::Hit

For example, we run
gaushit but not **fasthit** and
we only need the
protoDUNE disambiguation
hitpdune

**These are the Pandora
producers**

Important: things
downstream of Pandora
sometimes contain its
name if they use Pandora's
output, but these modules
are NOT part of Pandora



2) Let's look at the reco .fcl file

Small exercise: compare the list of producers inside reco: [...]

with the print out messages you have seen when running protoDUNE_reco.fcl

```
reco: [ rns,
        #optical hits and flashes
        ophit, opflash,
        #TPC wire signals
        caldata,
        #hit reconstruction
        gaushit, #fasthit,
        #space point solver
        reco3d,
        #real disambiguation
        hitpdune,
        # 3d dbcluster
        #dbcluster,
        #cluster reco
        linecluster,
        #feature labeling
        emtrkmichelid,
        #pandora
        pandora, pandoraTrack, pandoraShower,
        pandoracalo, pandorapid, pandoracali, pandoracalipid,
        #pmattrack
        pmtrack, pmtrackcalo, pmtrackpid, pmtrackcali, pmtrackcalipid
        #pmtrajfit, pmtrajfitcalo, pmtrajfitpid
        #shower reconstruction
        #blurredcluster, emshower, emshower3d, mergeemshower3d
    ]
```

Summary print out from the art:TimeTracker service

TimeTracker printout (sec)	Min	Avg	Max	Median	RMS	nEvts
Full event	10.9239	16.4927	30.7107	13.8305	6.03821	10
source:RootInput(read)	0.000281449	0.000515754	0.00146504	0.000427652	0.000323607	10
reco:rns:RandomNumberSaver	2.6142e-05	5.42986e-05	0.000274468	3.07545e-05	7.3424e-05	10
reco:ophit:OpHitFinder	0.000472357	0.0176818	0.168036	0.00110279	0.0501185	10
reco:opflash:OpFlashFinder	0.000192166	0.000496848	0.0023595	0.000310921	0.000622998	10
reco:caldata:DataPrepModule	8.44578	8.69662	10.8608	8.45596	0.721432	10
reco:gausshit:GausHitFinder	0.28158	0.480787	0.828892	0.455886	0.146557	10
reco:reco3d:SpacePointSolver	0.0127969	2.40228	17.5862	0.0844079	5.32705	10
reco:hitpdune:DisambigFromSpacePoints	0.011602	0.0208112	0.0369639	0.0189795	0.0081143	10
reco:linecluster:LineCluster	0.0125113	0.0201407	0.0328741	0.0186305	0.00655096	10
reco:emtrkmichelid:EmTrackMichelId	1.3143	2.51923	3.50061	2.6984	0.775743	10
reco:pandora:StandardPandora	0.179093	0.411216	0.670182	0.408648	0.146957	10
reco:pandoraTrack:LArPandoraTrackCreation	0.00202527	0.00458205	0.00726229	0.00490643	0.00181189	10
reco:pandoraShower:LArPandoraShowerCreation	0.000309023	0.000727246	0.0019323	0.000435659	0.000519239	10
reco:pandoracalo:Calorimetry	0.00116318	0.00648228	0.0164835	0.0042571	0.0052587	10
reco:pandorapid:Chi2ParticleID	0.000159165	0.00107007	0.00864384	0.0023664	0.00252532	10
reco:pandoracali:CalibrationEdXPDSP	0.00050625	0.651673	6.5022	0.00184645	1.95018	10
reco:pandoracalipid:Chi2ParticleID	0.000110441	0.000182945	0.000264772	0.000183379	5.83571e-05	10
reco:pmtrack:PMAlgTrackMaker	0.149092	1.18471	4.11319	0.815881	1.18543	10
reco:pmtrackcalo:Calorimetry	6.4012e-05	0.00387695	0.0141003	0.00107946	0.00518883	10
reco:pmtrackpid:Chi2ParticleID	5.6334e-05	0.000168095	0.000272963	0.000165131	5.80321e-05	10
reco:pmtrackcali:CalibrationEdXPDSP	4.3335e-05	0.00101092	0.00232963	0.000639647	0.000922123	10
reco:pmtrackcalipid:Chi2ParticleID	3.0099e-05	0.000114284	0.000223515	0.000114486	5.80596e-05	10
reco:TriggerResults:TriggerResultInserter	1.079e-05	1.18981e-05	1.8133e-05	1.12365e-05	2.09407e-06	10
end_path:out1:RootOutput	1.276e-06	1.5665e-06	2.907e-06	1.43e-06	4.54485e-07	10
end_path:out1:RootOutput(write)	0.0282791	0.0680769	0.114847	0.0688115	0.0321029	10

**In particular, how much time does it take to run them?
(use TimeTracker info)**



3) Some more info about Pandora producers

```
emacs -nw $DUNETPC_DIR/job/pandoramodules_dune.fcl
```

And open it!

```
dune_pandora:  
{  
    module_type:  
    GeantModuleLabel:  
    HitFinderModuleLabel:  
    EnableMCParticles:  
    EnableProduction:  
    EnableLineGaps:  
    UseGlobalCoordinates:  
    UseHitWidths:  
    ShouldRunAllHitsCosmicReco:  
    ShouldRunStitching:  
    ShouldRunCosmicHitRemoval:  
    ShouldRunSlicing:  
    ShouldRunNeutrinoRecoOption:  
    ShouldRunCosmicRecoOption:  
    ShouldPerformSliceId:  
    PrintOverallRecoStatus:  
}
```

```
"StandardPandora"  
"largeant"  
"linecluster"  
false  
true  
true  
true  
false  
false  
false  
false  
false  
false  
false  
false  
false  
false
```

General default options for
Pandora DUNE

Important: Notice that the
only input to Pandora is a
collection of hits. Although
linecluster produces also
clusters, Pandora won't use
them, but run its own
clustering algorithms

And protoDUNE
specific setup

```
protodune_pandora:  
protodune_pandora.ConfigFile:  
protodune_pandora.ShouldRunAllHitsCosmicReco:  
protodune_pandora.ShouldRunStitching:  
protodune_pandora.ShouldRunCosmicHitRemoval:  
protodune_pandora.ShouldRunSlicing:  
protodune_pandora.ShouldRunNeutrinoRecoOption:  
protodune_pandora.ShouldRunCosmicRecoOption:  
protodune_pandora.ShouldPerformSliceId:
```

```
@local::dune_pandora  
"PandoraSettings_Master_ProtoDUNE.xml"  
true  
true  
true  
true  
true  
true  
true
```

This is the .xml settings file
containing the algorithms
that will be run

Notice that, unlike DUNE FD, protoDUNE will have
cosmic rays, so corresponding options for cosmic
treatment are turned on



4) Identify output products

Now let's have a look at what the reco output file contains

Run! `lar -c eventdump.fcl -n 1 protoDUNE_pion_2GeV_mono_G4_detsim_events_reco.root`

PROCESS NAME	MODULE_LABEL..	PRODUCT INSTANCE NAME..	DATA PRODUCT TYPE.....	SIZE
SinglesGen..	rns.....	std::vector<art::RNGsnapshot>.....1
SinglesGen..	TriggerResults	art::TriggerResults.....-
SinglesGen..	generator.....	std::vector<simb::MCTruth>.....1
G4.....	rns.....	std::vector<art::RNGsnapshot>.....2
G4.....	TriggerResults	art::TriggerResults.....-
G4.....	largeant.....	std::vector<sim::OpDetBacktrackerRecord>.....59
G4.....	largeant.....	std::vector<simb::MCParticle>.....	..630
G4.....	largeant.....	std::vector<sim::AuxDetSimChannel>.....	.2048
G4.....	largeant.....	art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo>.....	..630
G4.....	largeant.....	std::vector<sim::SimChannel>.....	.2188
G4.....	largeant.....	std::vector<sim::SimPhotonsLite>.....59
Detsim.....	rns.....	std::vector<art::RNGsnapshot>.....1
Detsim.....	TriggerResults	art::TriggerResults.....-
Detsim.....	opdigi.....	std::vector<raw::OpDetWaveform>.....	..132
Detsim.....	daq.....	std::vector<raw::RawDigit>.....	15360
Detsim.....	crt.....	art::Assns<sim::AuxDetSimChannel,CRT::Trigger,void>.....0
Detsim.....	crt.....	std::vector<CRT::Trigger>.....0
Detsim.....	opdigi.....	std::vector<sim::OpDetDivRec>.....59

Detsim products

Unless you choose to drop products by saying so in your .fcl files, your output root file will add products to the existing ones, so you will have all the products of processes run continues overleaf...



4) Identify output products

Now let's have a look at what the output file contains

Run! `lar -c eventdump.fcl -n 1 protoDUNE_pion_2GeV_mono_G4_detsim_events_reco.root`

Some of the reco products

PROCESS NAME	MODULE_LABEL..	PRODUCT INSTANCE NAME..	DATA PRODUCT TYPE.....	.SIZE
Reco.....	TriggerResults	art::TriggerResults.....
Reco.....	gaushit.....	art::Assns<raw::RawDigit,recob::Hit,void>.....	.1058
Reco.....	gaushit.....	std::vector<recob::Hit>.....	.1058
Reco.....	gaushit.....	art::Assns<recob::Wire,recob::Hit,void>.....	.1058
Reco.....	hitpdune.....	art::Assns<recob::Hit,recob::SpacePoint,void>.....	.1584
Reco.....	hitpdune.....	std::vector<recob::Hit>.....	.1058
Reco.....	hitpdune.....	art::Assns<raw::RawDigit,recob::Hit,void>.....	.1058
Reco.....	hitpdune.....	art::Assns<recob::Wire,recob::Hit,void>.....	.1058
Reco.....	linecluster...	art::Assns<recob::Cluster,recob::Hit,void>.....	.718
Reco.....	linecluster...	std::vector<recob::EndPoint2D>.....	..0
Reco.....	linecluster...	std::vector<recob::Cluster>.....	.39
Reco.....	linecluster...	art::Assns<recob::Cluster,recob::EndPoint2D,unsigned short>.....	..0
Reco.....	linecluster...	art::Assns<recob::Cluster,recob::Vertex,unsigned short>.....	..0
Reco.....	linecluster...	art::Assns<recob::Wire,recob::Hit,void>.....	.1054
Reco.....	linecluster...	std::vector<recob::Vertex>.....	..0
Reco.....	linecluster...	std::vector<recob::Hit>.....	.1054

Some of the producers names
you have seen before

This is the only input for Pandora
(collection of hits)

continues overleaf...

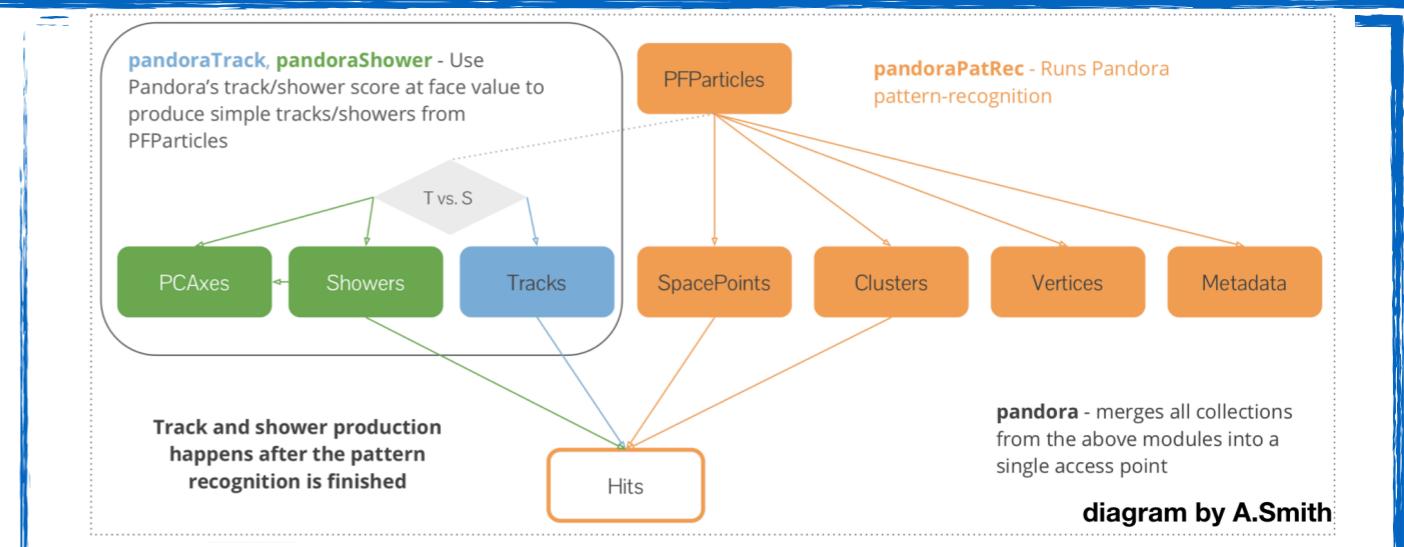


4) Identify output products

Pandora products

PROCESS NAME	MODULE_LABEL..	PRODUCT INSTANCE NAME..	DATA PRODUCT TYPE.....	.SIZE
Reco.....	pandora.....	std::vector<recob::PFParticle>.....	... 12
Reco.....	pandora.....	std::vector<recob::Vertex>.....	... 11
Reco.....	pandora.....	std::vector<larpandoraobj::PFParticleMetadata>.....	... 12
Reco.....	pandora.....	std::vector<recob::SpacePoint>.....	.. 508
Reco.....	pandora.....	std::vector<recob::Cluster>.....	... 3
Reco.....	pandora.....	std::vector<anab::T0>.....	... 0
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::SpacePoint, void>.....	.. 508
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::Vertex, void>.....	... 11
Reco.....	pandora.....	art::Assns<recob::Cluster, recob::Hit, void>.....	.. 620
Reco.....	pandora.....	art::Assns<recob::PFParticle, recob::Cluster, void>.....	... 33
Reco.....	pandora.....	art::Assns<recob::PFParticle, larpandoraobj::PFParticleMetadata, void>.....	... 12
Reco.....	pandora.....	art::Assns<recob::PFParticle, anab::T0, void>.....	... 0
Reco.....	pandora.....	art::Assns<recob::SpacePoint, recob::Hit, void>.....	.. 508
Reco.....	pandoraTrack..	std::vector<recob::Track>.....	... 7
Reco.....	pandoraTrack..	art::Assns<recob::PFParticle, recob::Track, void>.....	... 7
Reco.....	pandoraTrack..	art::Assns<recob::Track, recob::Hit, void>.....	.. 345
Reco.....	pandoraTrack..	art::Assns<recob::Track, recob::Hit, recob::TrackHitMeta>.....	.. 345
Reco.....	pandoraShower..	std::vector<recob::Shower>.....	... 4
Reco.....	pandoraShower..	std::vector<recob::PCAxiS>.....	... 4
Reco.....	pandoraShower..	art::Assns<recob::Shower, recob::Hit, void>.....	.. 257
Reco.....	pandoraShower..	art::Assns<recob::PFParticle, recob::PCAxiS, void>.....	... 4
Reco.....	pandoraShower..	art::Assns<recob::PFParticle, recob::Shower, void>.....	... 4
Reco.....	pandoraShower..	art::Assns<recob::Shower, recob::PCAxiS, void>.....	... 4

Note: sadly your products won't be
Nicely ordered like this...



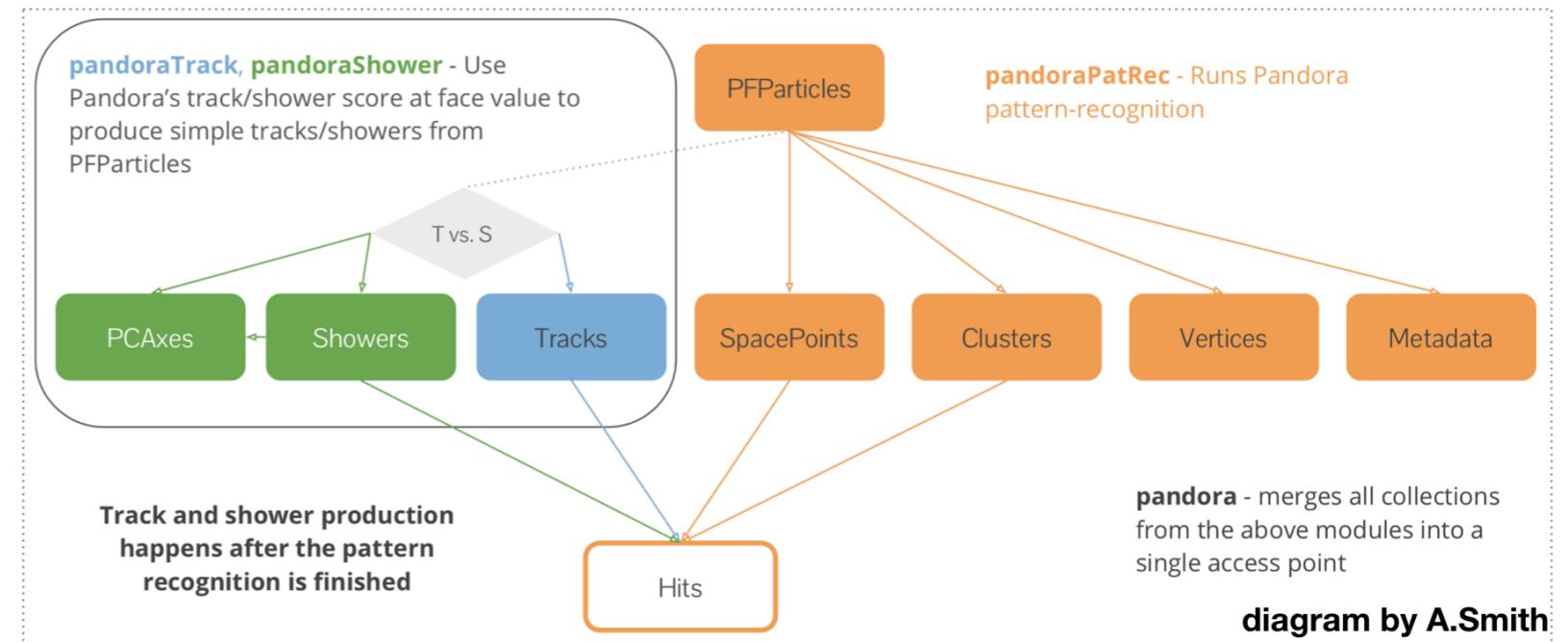


4) Identify output products

Legend:

Product

= Assns



Art::Associations (more info [here](#) and [here](#))

They are a way to make references between two data objects stored in the event. LArSoft provides a set of in-line functions to create and retrieve these associations. For example, if I have in my event:

```
Reco..... | pandora..... | ..... | std::vector<recob::Cluster>..... | ...3  
Reco..... | pandora..... | ..... | art::Assns<recob::Cluster, recob::Hit, void>..... | ..620
```

I can use the **art::Assns<recob::Cluster, recob::Hit>** to, for each cluster in the **std::vector<recob::Cluster>**, find the **<recob::Hit> objects** that have been associated to such cluster; in this case it means the recob::Hits in the recob::Cluster

Small exercise: find all the associations of the diagram above in your output



4) Identify output products

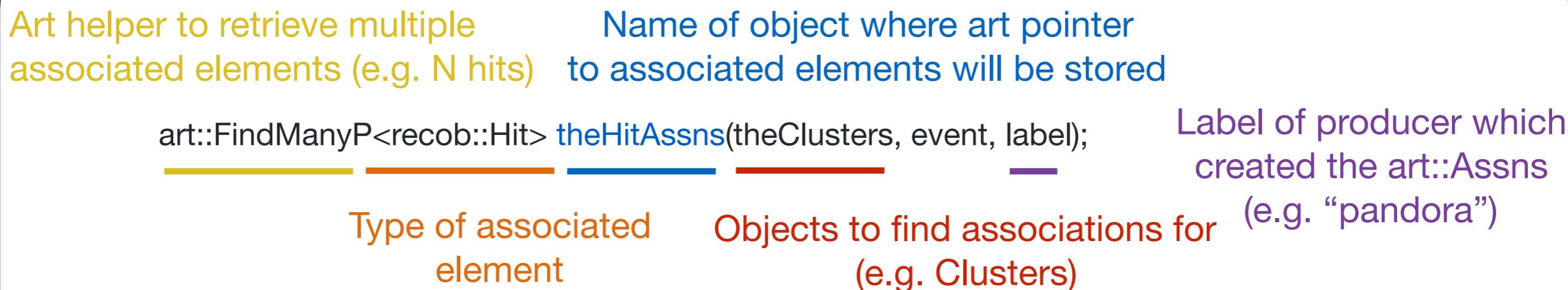
Art::Associations (more info [here](#) and [here](#))

They are a way to make references between two data objects stored in the event. LArSoft provides a set of in-line functions to create and retrieve these associations. For example, if I have in my event:

```
Reco..... | pandora..... | ..... | std::vector<recob::Cluster>..... | ....3  
Reco..... | pandora..... | ..... | art::Assns<recob::Cluster, recob::Hit, void>..... | ...620
```

I can retrieve the **art::Assns<recob::Cluster, recob::Hit>** using the [FindOne/FindMany](#) helpers

The syntax would be something like this:



Small exercise: have a look at the functions in [larpandora/LArPandoraHelper.cxx](#) and find the one handling the Cluster-Hit associations for you



Visualise different stages of the reconstruction in events with cosmics



Consolidated Output

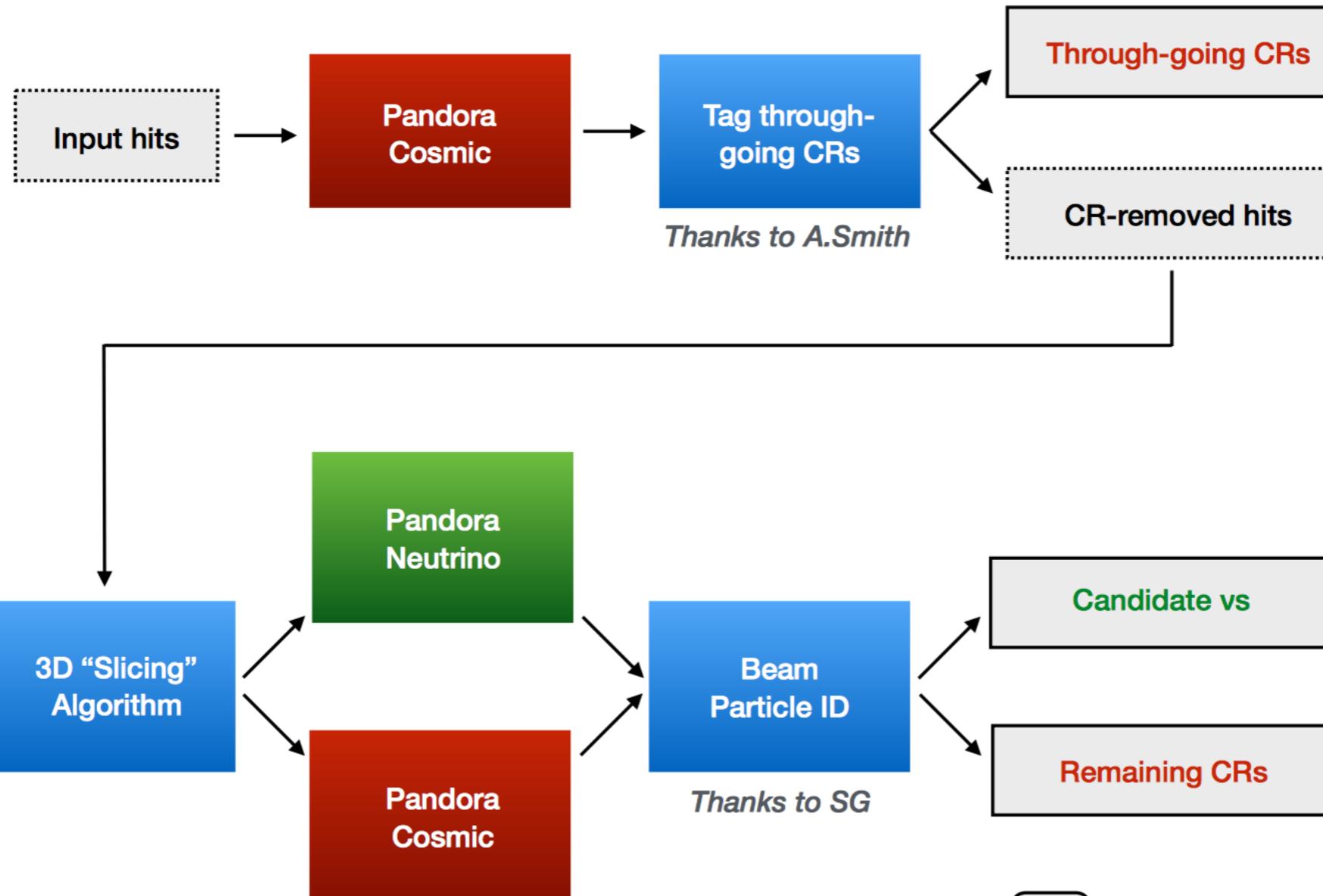
UNIVERSITY OF
CAMBRIDGE

Consolidated Reconstruction: Overview

DUNE
DEEP U
NEUTRINO

Note: The events you generated have no cosmics, but still we will run the master settings file

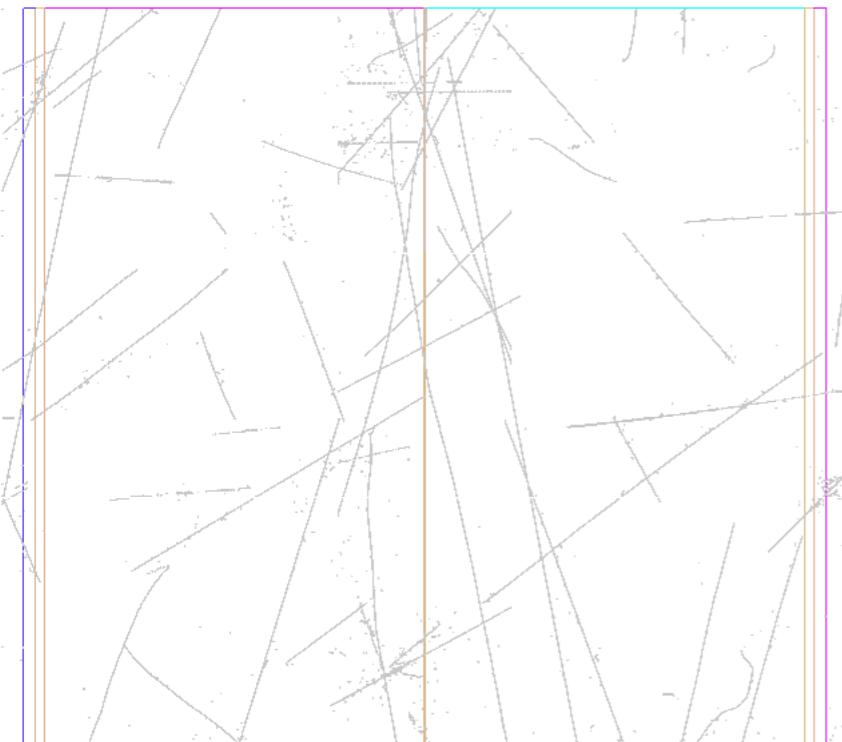
Thanks to J.Marshall



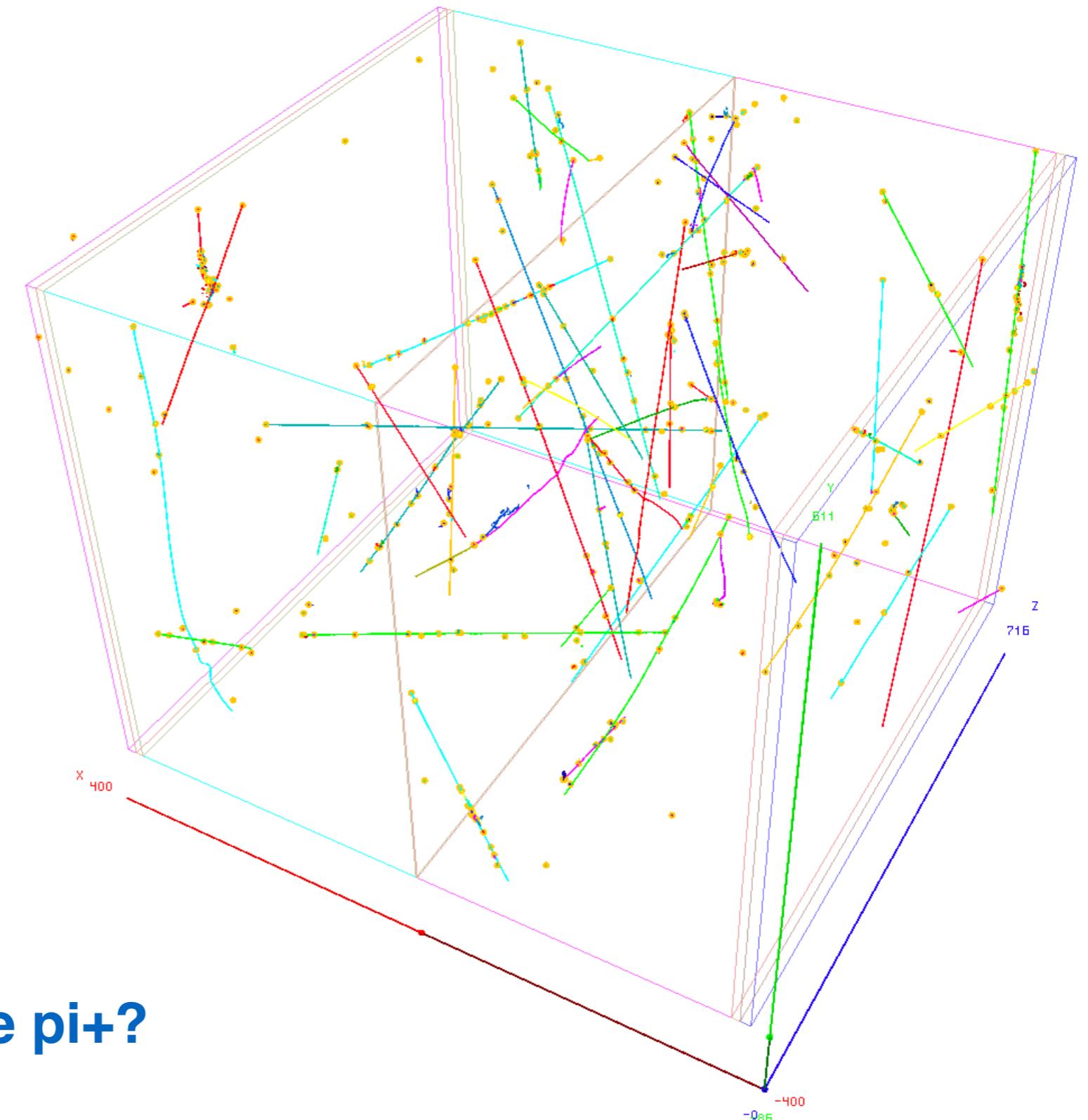


Visualize more events

Collection plane hits



Cosmic events are much busier!



Can you find the π^+ ?



Visualize more events

We can visualise the reconstruction at various stages

emacs -nw MyPandoraSettings_Master_ProtoDUNE.xml

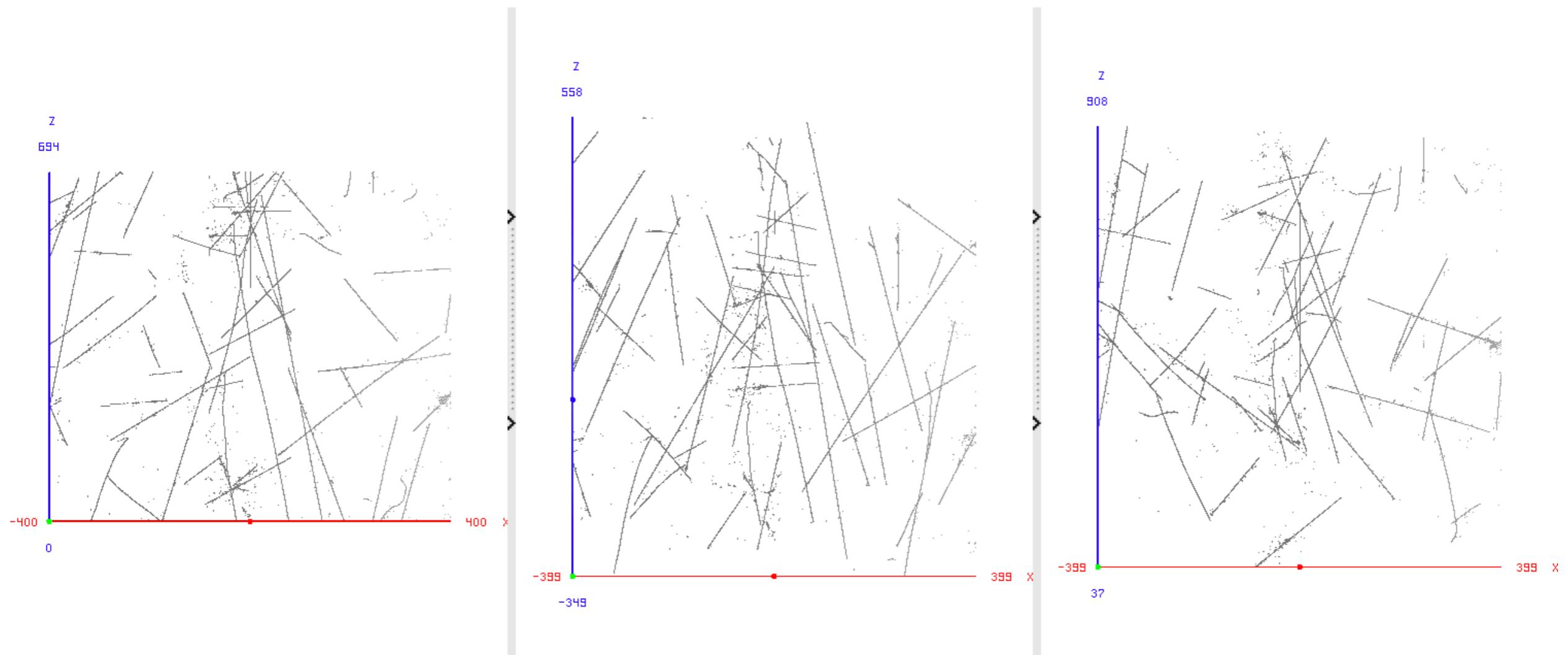
```
<algorithm type = "LArMaster">
    <CRSettingsFile>PandoraSettings_Cosmic_Standard.xml</CRSettingsFile>
    <NuSettingsFile>PandoraSettings_TestBeam_ProtoDUNE.xml</NuSettingsFile>
    <SlicingSettingsFile>PandoraSettings_Slicing_ProtoDUNE.xml</SlicingSettingsFile>
    <StitchingTools>
        <tool type = "LArStitchingCosmicRayMerging"><ThreeDStitchingMode>true</
ThreeDStitchingMode></tool>
        <tool type = "LArStitchingCosmicRayMerging"><ThreeDStitchingMode>false</
ThreeDStitchingMode></tool>
    </StitchingTools>
    <CosmicRayTaggingTools>
        <tool type = "LArCosmicRayTagging"/>
    </CosmicRayTaggingTools>
    <SliceIdTools>
        <tool type = "LArBdtBeamParticleId">
            <BdtName>BeamParticleId</BdtName>
            <BdtFileName>PandoraBdt_v03_14_00.xml</BdtFileName>
            <MinAdaBDTScore>-0.225</MinAdaBDTScore>
        </tool>
    </SliceIdTools>
    <InputHitListName>Input</InputHitListName>
    <RecreatedPfoListName>RecreatedPfos</RecreatedPfoListName>
    <RecreatedClusterListName>RecreatedClusters</RecreatedClusterListName>
    <RecreatedVertexListName>RecreatedVertices</RecreatedVertexListName>
    <VisualizeOverallRecostatus>true</VisualizeOverallRecostatus>
</algorithm>
```



Visualize more events

We can visualise the reconstruction at various stages

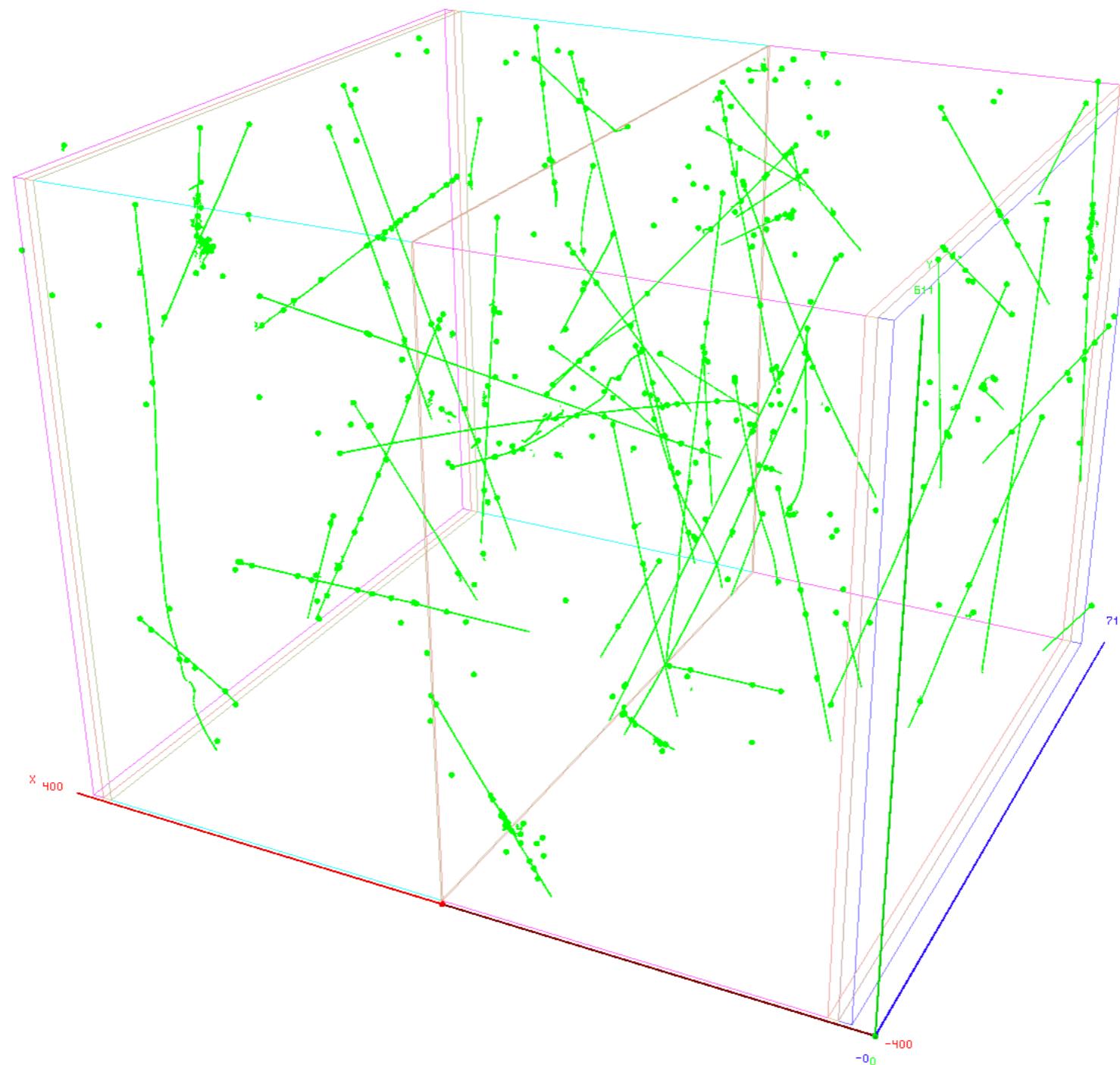
1- Input 2D hits





Visualize more events

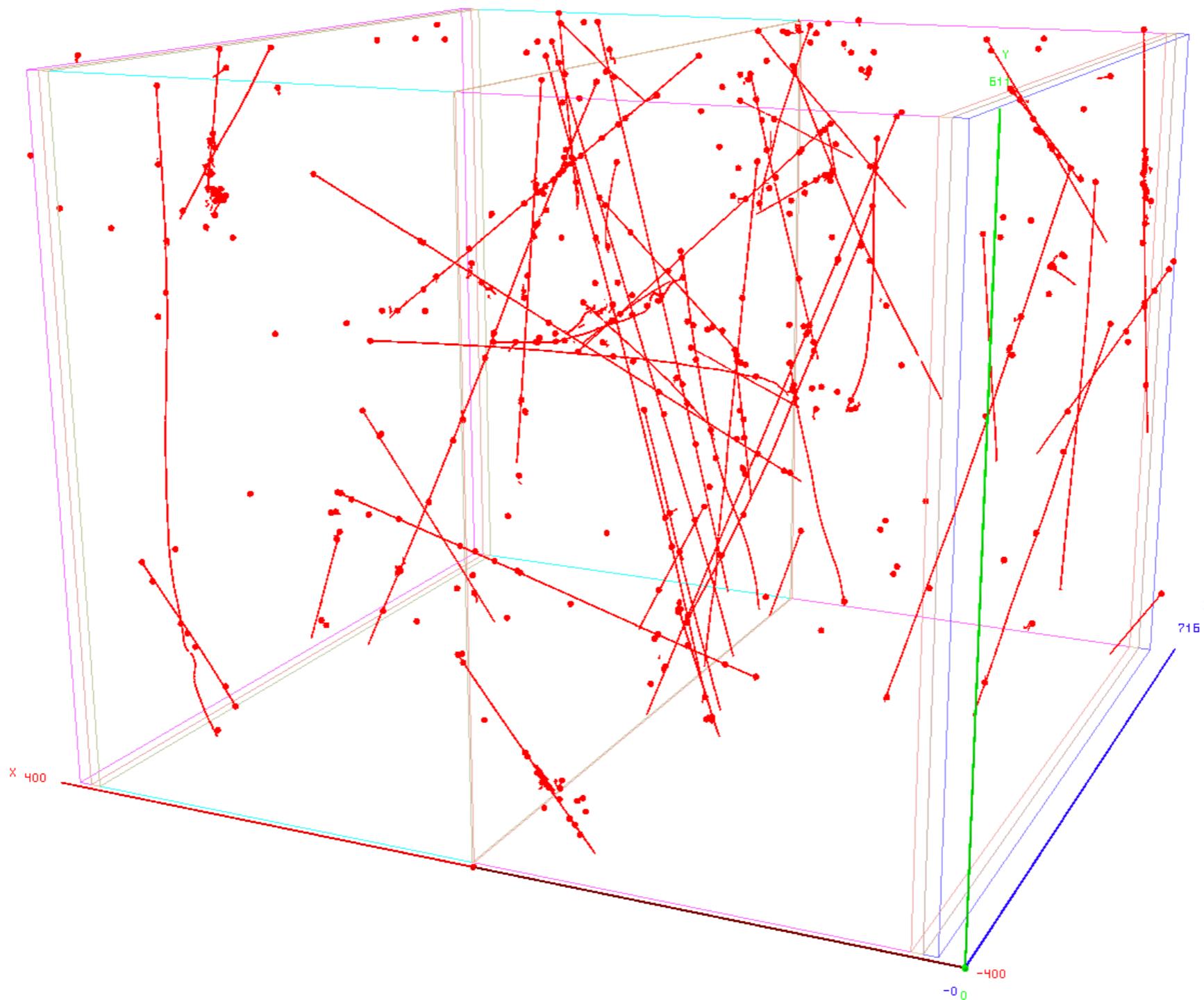
2 - Cosmic Ray reconstruction





Visualize more events

3 - Stitching



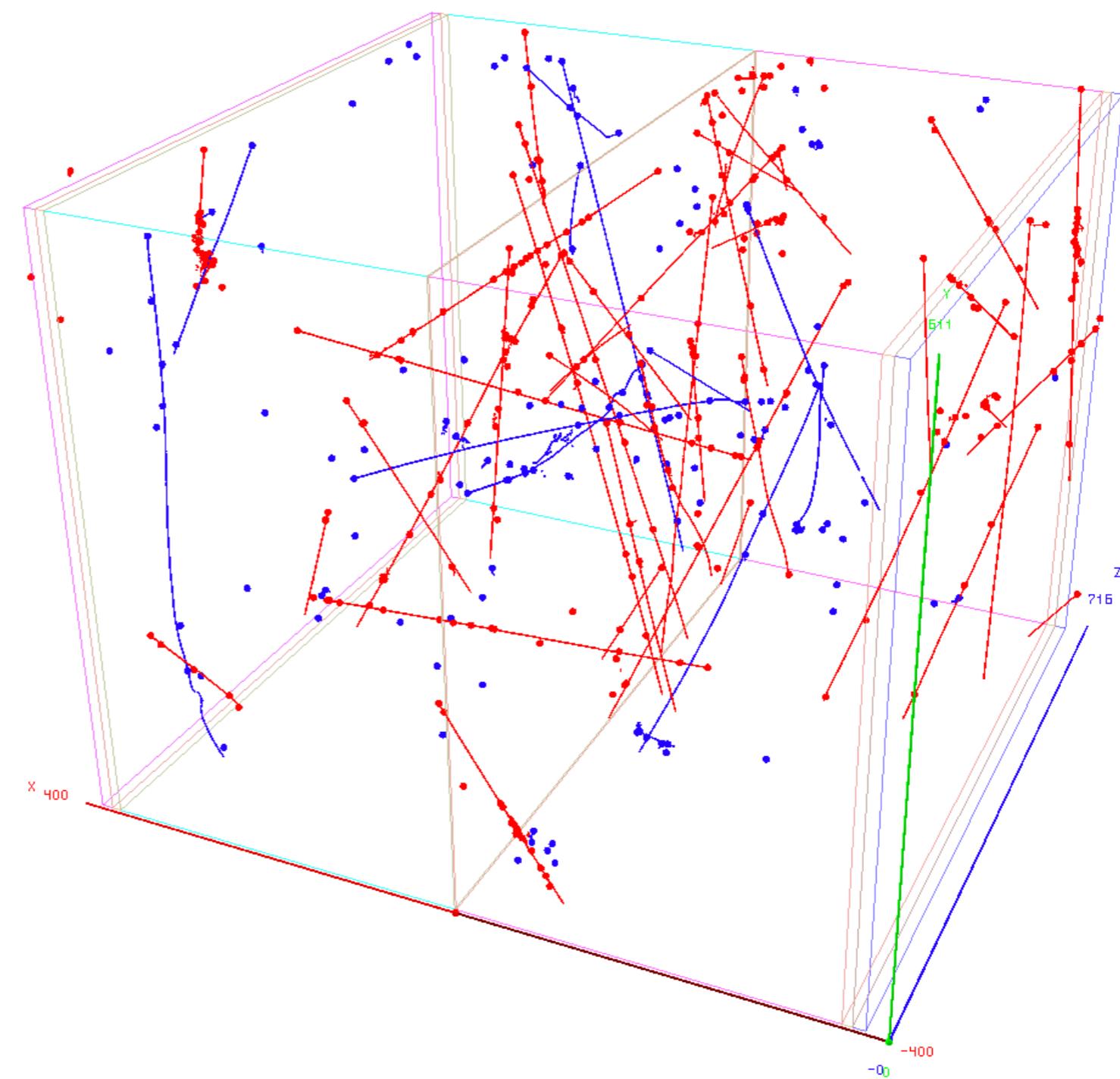


Visualize more events

4 - Clear Cosmic Rays tagged

**Clear cosmics:
Tagged!**

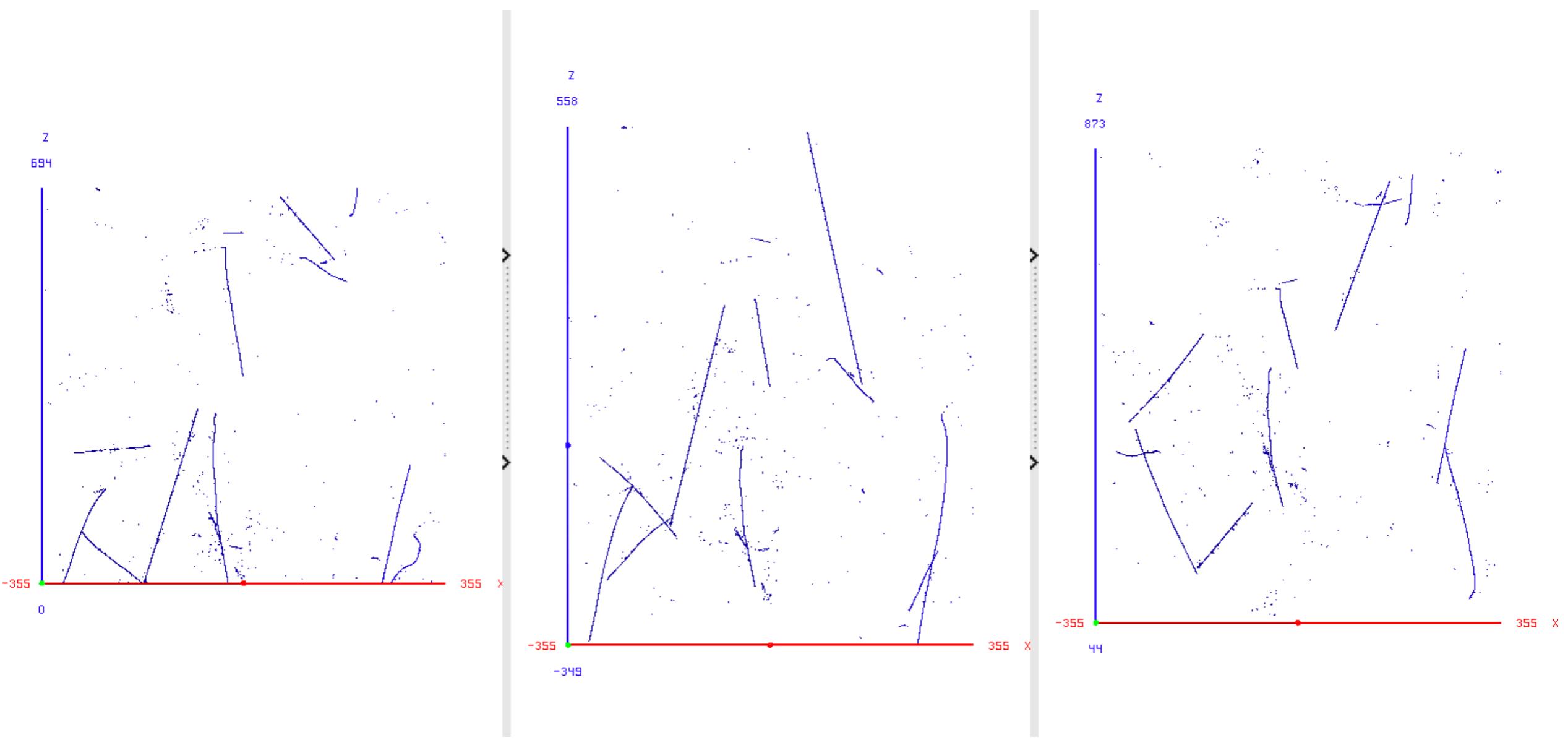
**Ambiguous
cosmics:
they go to
next stage**





Visualize more events

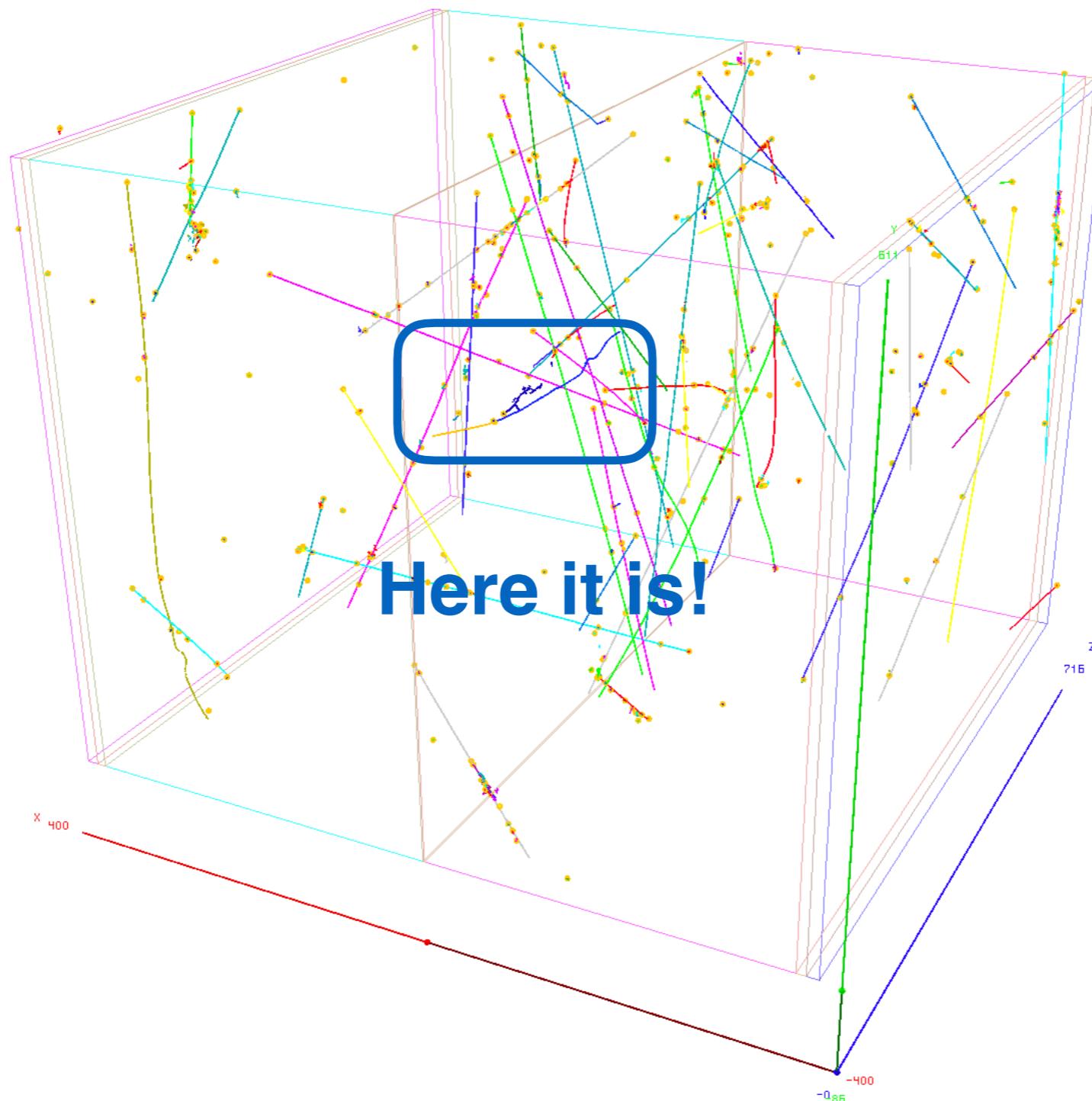
5 - Hits remaining after CR tagging - each PFO represents a slice





Visualize more events

6 - Final collection of particles





**See which algorithms are running
and validate against MC truth**



See which algorithms are running

```
cp $DUNETPC_DIR/scripts/PandoraSettings_TestBeam_ProtoDUNE.xml MyPandoraSettings_TestBeam_ProtoDUNE.xml
```

1) Edit MyPandoraSettings_TestBeam_ProtoDUNE.xml

```
<!-- GLOBAL SETTINGS -->
<IsMonitoringEnabled>true</IsMonitoringEnabled>
<ShouldDisplayAlgorithmInfo>true</ShouldDisplayAlgorithmInfo>
<SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>
```

Note: You might want to set IsMonitoringEnabled to false in MyPandoraSettings_Master_ProtoDUNE.xml to avoid the visualisation (up to you!)



See which algorithms are running

```
cp $DUNETPC_DIR/scripts/PandoraSettings_TestBeam_ProtoDUNE.xml MyPandoraSettings_TestBeam_ProtoDUNE.xml
```

1) Edit MyPandoraSettings_TestBeam_ProtoDUNE.xml

```
<!-- GLOBAL SETTINGS -->
<IsMonitoringEnabled>true</IsMonitoringEnabled>
<ShouldDisplayAlgorithmInfo>true</ShouldDisplayAlgorithmInfo>
<SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>
```

2) Tell your .fcl file to use the local version

```
cp $DUNETPC_DIR/job/protoDUNE_reco.fcl myprotoDUNE_reco.fcl
```

```
emacs -nw myprotoDUNE_reco.fcl (or your favourite editor)
```

and add:

```
#Pandora configurations
physics.producers.pandora.HitFinderModuleLabel: "linecluster"
physics.producers.pandora.ConfigFile: "MyPandoraSettings_Master_ProtoDUNE.xml"
```



See which algorithms are running

```
cp $DUNETPC_DIR/scripts/PandoraSettings_TestBeam_ProtoDUNE.xml MyPandoraSettings_TestBeam_ProtoDUNE.xml
```

1) Edit MyPandoraSettings_TestBeam_ProtoDUNE.xml

```
<!-- GLOBAL SETTINGS -->
<IsMonitoringEnabled>true</IsMonitoringEnabled>
<ShouldDisplayAlgorithmInfo>true</ShouldDisplayAlgorithmInfo>
<SingleHitTypeClusteringMode>true</SingleHitTypeClusteringMode>
```

2) Tell your .fcl file to use the local version

```
cp $DUNETPC_DIR/job/protoDUNE_reco.fcl myprotoDUNE_reco.fcl
```

```
emacs -nw myprotoDUNE_reco.fcl (or your favourite editor)
```

and add:

```
#Pandora configurations
physics.producers.pandora.HitFinderModuleLabel: "linecluster"
physics.producers.pandora.ConfigFile: "MyPandoraSettings_Master_ProtoDUNE.xml"
```

3) Tell your master file to use the local version

```
emacs -nw MyPandoraSettings_Master_ProtoDUNE.xml to use the local TestBeam one:
```

```
<NuSettingsFile>MyPandoraSettings_TestBeam_ProtoDUNE.xml</NuSettingsFile>
```

Note: This will only print out the algorithms of the TestBeam pass, but not the other processes (cosmic ray, slicing)



See which algorithms are running

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

```
> Running Algorithm: Alg0001, LArPreProcessing  
> Running Algorithm: Alg0002, LArClusteringParent  
----> Running Algorithm: Alg0003, LArTrackClusterCreation  
> Running Algorithm: Alg0004, LArLayerSplitting  
> Running Algorithm: Alg0005, LArLongitudinalAssociation  
> Running Algorithm: Alg0006, LArTransverseAssociation  
> Running Algorithm: Alg0007, LArLongitudinalExtension  
> Running Algorithm: Alg0008, LArTransverseExtension  
> Running Algorithm: Alg0009, LArCrossGapsAssociation  
> Running Algorithm: Alg0010, LArCrossGapsExtension  
> Running Algorithm: Alg0011, LArOvershootSplitting  
> Running Algorithm: Alg0012, LArBranchSplitting  
> Running Algorithm: Alg0013, LArKinkSplitting  
> Running Algorithm: Alg0014, LArTrackConsolidation  
> Running Algorithm: Alg0016, LArClusteringParent  
----> Running Algorithm: Alg0017, LArTrackClusterCreation  
> Running Algorithm: Alg0018, LArLayerSplitting  
> ...  
> Running Algorithm: Alg0028, LArTrackConsolidation  
> Running Algorithm: Alg0030, LArClusteringParent  
----> Running Algorithm: Alg0031, LArTrackClusterCreation  
> Running Algorithm: Alg0032, LArLayerSplitting  
> ...  
> Running Algorithm: Alg0042, LArTrackConsolidation  
> Running Algorithm: Alg0044, LArCandidateVertexCreation  
> Running Algorithm: Alg0045, LArEnergyKickVertexSelection  
----> Running Algorithm Tool: Tool0001, LArEnergyKickFeature  
----> Running Algorithm Tool: Tool0002, LArLocalAsymmetryFeature
```

And now you should see the information of the algorithms running!

PreProcessing

2D Reconstruction

Note: sets of algorithms are run three times, one per view (U,V,W)

Vertex Algorithms

Some algorithms run tools, also in the order described in the settings files. They might run the tools multiple types, e.g. here to compute features



See which algorithms are running

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

```
> Running Algorithm: Alg0047, LArThreeDTransverseTracks
----> Running Algorithm Tool: Tool0003, LArClearTracks
----> Running Algorithm Tool: Tool0003, LArClearTracks
----> Running Algorithm Tool: Tool0004, LArLongTracks
----> Running Algorithm Tool: Tool0005, LArOvershootTracks
----> Running Algorithm Tool: Tool0006, LArUndershootTracks
----> Running Algorithm Tool: Tool0007, LArOvershootTracks
----> Running Algorithm Tool: Tool0008, LArUndershootTracks
----> Running Algorithm Tool: Tool0003, LArClearTracks
----> Running Algorithm Tool: Tool0004, LArLongTracks
----> Running Algorithm Tool: Tool0005, LArOvershootTracks
----> Running Algorithm Tool: Tool0006, LArUndershootTracks
----> Running Algorithm Tool: Tool0007, LArOvershootTracks
----> Running Algorithm Tool: Tool0008, LArUndershootTracks
----> Running Algorithm Tool: Tool0009, LArMissingTrackSegment
----> Running Algorithm Tool: Tool0010, LArTrackSplitting
----> Running Algorithm Tool: Tool0011, LArLongTracks
----> Running Algorithm Tool: Tool0003, LArClearTracks
----> Running Algorithm Tool: Tool0004, LArLongTracks
----> Running Algorithm Tool: Tool0005, LArOvershootTracks
----> Running Algorithm Tool: Tool0006, LArUndershootTracks
----> Running Algorithm Tool: Tool0007, LArOvershootTracks
----> Running Algorithm Tool: Tool0008, LArUndershootTracks
----> Running Algorithm Tool: Tool0009, LArMissingTrackSegment
----> Running Algorithm Tool: Tool0010, LArTrackSplitting
----> Running Algorithm Tool: Tool0011, LArLongTracks
----> Running Algorithm Tool: Tool0012, LArTracksCrossingGaps
----> Running Algorithm Tool: Tool0013, LArMissingTrack
> Running Algorithm: Alg0048, LArThreeDLongitudinalTracks
...
...
```

3D Reconstruction

Note: the algorithm tools have a specific ordering and, if any tool makes a change, the *rank-three tensor* (in which the suitability of all combinations of clusters is stored) is updated and the full list of tools runs again until no more changes are made

And now you should see the information of the algorithms running!

+ rest of 3D algorithms (showers, 3Dhits, particle recovery...)



and validate against MC truth

We are now going to use the EventValidation algorithm in Pandora to print the reco-true matching you have seen in the previous tutorial

1) Edit `MyPandoraSettings_Master_ProtoDUNE.xml` and add a call to the EventValidation algorithm at the end (copy this in your file, just before the end `</pandora>`)

```
<algorithm type = "LArEventValidation">
    <CaloHitListName>CaloHitList2D</CaloHitListName>
    <MCParticleListName>Input</MCParticleListName>
    <PfoListName>RecreatedPfos</PfoListName>
    <UseTrueNeutrinosOnly>false</UseTrueNeutrinosOnly>
    <PrintAllToScreen>false</PrintAllToScreen>
    <PrintMatchingToScreen>true</PrintMatchingToScreen>
    <WriteToTree>false</WriteToTree>
    <OutputTree>Validation</OutputTree>
    <OutputFile>Validation.root</OutputFile>
    <TestBeamMode>true</TestBeamMode>
</algorithm>
```

2) Edit `myprotoDUNE_reco.fcl` to enable MC Particles (truth information) and check you are using `MyPandoraSettings_Master_ProtoDUNE.xml`

```
physics.producers.pandora.ConfigFile:
physics.producers.pandora.EnableMCParticles:
```

```
"MyPandoraSettings_Master_ProtoDUNE.xml"
true
```



and validate against MC truth

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

Now you will be printing this type of information:

---INTERPRETED-MATCHING-OUTPUT---

BEAM_PARTICLE_PI_PLUS (Nuance 2000, Nu 0, TB 1, CR 0)

protoDUNE

IsCorrectTB (NNuMatches: 1)

PrimaryId 1, Nu 0, TB 1, CR 0, MCPDG 211, Energy 1.98644, Dist. 47.1243, nMCHits 1302 (442, 422, 438)

-MatchedPfoId 1, Nu 0, TB 1, CR 0, PDG 211, nMatchedHits 1116 (372, 320, 424), nPfoHits 1116 (372, 320, 424)

---SUMMARY---

#CorrectTB: 1/1, Fraction: 1

---INTERPRETED-MATCHING-OUTPUT---

CCRES_E_P_PIZERO (Nuance 1004, Nu 1, TB 0, CR 0)

DUNE FD

IsCorrectNu (NNuMatches: 4)

PrimaryId 1, Nu 1, TB 0, CR 0, MCPDG 11, Energy 1.828, Dist. 43.3081, nMCHits 1712 (671, 417, 624)

-MatchedPfoId 1, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 1621 (657, 377, 587), nPfoHits 1677 (683, 383, 611)

PrimaryId 2, Nu 1, TB 0, CR 0, MCPDG 2212, Energy 1.89848, Dist. 8.45417, nMCHits 448 (141, 125, 182)

-MatchedPfoId 2, Nu 1 [NuId: 1], TB 0, CR 0, PDG 13, nMatchedHits 407 (122, 116, 169), nPfoHits 444 (124, 131, 189)

PrimaryId 3, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.45388, Dist. 22.4647, nMCHits 442 (102, 198, 142)

-MatchedPfoId 3, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 397 (93, 189, 115), nPfoHits 398 (93, 190, 115)

PrimaryId 4, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.178079, Dist. 30.5587, nMCHits 211 (30, 100, 81)

-MatchedPfoId 4, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 85 (13, 41, 31), nPfoHits 87 (13, 43, 31)

---SUMMARY---

#CorrectNu: 1/1, Fraction: 1



and validate against MC truth

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

Now you will be printing this type of information:

```
---INTERPRETED-MATCHING-OUTPUT----- DUNE FD
CCRES_E_P_PIZERO (Nuance 1004, Nu 1, TB 0, CR 0)
IsCorrectNu (NNuMatches: 4)
PrimaryId 1, Nu 1, TB 0, CR 0, MCPDG 11, Energy 1.828, Dist. 43.3081, nMCHits 1712 (671, 417, 624)
-MatchedPfoId 1, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 1621 (657, 377, 587), nPfoHits 1677 (683, 383, 611)
PrimaryId 2, Nu 1, TB 0, CR 0, MCPDG 2212, Energy 1.89848, Dist. 8.45417, nMCHits 448 (141, 125, 182)
-MatchedPfoId 2, Nu 1 [NuId: 1], TB 0, CR 0, PDG 13, nMatchedHits 407 (122, 116, 169), nPfoHits 444 (124, 131, 189)
PrimaryId 3, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.45388, Dist. 22.4647, nMCHits 442 (102, 198, 142)
-MatchedPfoId 3, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 397 (93, 189, 115), nPfoHits 398 (93, 190, 115)
PrimaryId 4, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.178079, Dist. 30.5587, nMCHits 211 (30, 100, 81)
-MatchedPfoId 4, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 85 (13, 41, 31), nPfoHits 87 (13, 43, 31)

---SUMMARY-----
#CorrectNu: 1/1, Fraction: 1
```

Primary true MC target (reconstructable) particles



and validate against MC truth

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

Now you will be printing this type of information:

```
---INTERPRETED-MATCHING-OUTPUT----- DUNE FD
CCRES_E_P_PIZERO (Nuance 1004, Nu 1, TB 0, CR 0)
IsCorrectNu (NNuMatches: 4)
PrimaryId 1, Nu 1, TB 0, CR 0, MCPDG 11, Energy 1.828, Dist. 43.3081, nMCHits 1712 (671, 417, 624)
-MatchedPfoId 1, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 1621 (657, 377, 587), nPfoHits 1677 (683, 383, 611)
PrimaryId 2, Nu 1, TB 0, CR 0, MCPDG 2212, Energy 1.89848, Dist. 8.45417, nMCHits 448 (141, 125, 182)
-MatchedPfoId 2, Nu 1 [NuId: 1], TB 0, CR 0, PDG 13, nMatchedHits 407 (122, 116, 169), nPfoHits 444 (124, 131, 189)
PrimaryId 3, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.45388, Dist. 22.4647, nMCHits 442 (102, 198, 142)
-MatchedPfoId 3, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 397 (93, 189, 115), nPfoHits 398 (93, 190, 115)
PrimaryId 4, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.178079, Dist. 30.5587, nMCHits 211 (30, 100, 81)
-MatchedPfoId 4, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 85 (13, 41, 31), nPfoHits 87 (13, 43, 31)

---SUMMARY-----
#CorrectNu: 1/1, Fraction: 1
```

Primary true MC target (reconstructable) particles
Interaction type, according to final state target MC particles



and validate against MC truth

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

Now you will be printing this type of information:

```
---INTERPRETED-MATCHING-OUTPUT--- DUNE FD
CCRES_E_P_PIZERO (Nuance 1004, Nu 1, TB 0, CR 0)
IsCorrectNu (NNuMatches: 4)
PrimaryId 1, Nu 1, TB 0, CR 0, MCPDG 11, Energy 1.828, Dist. 43.3081, nMCHits 1712 (671, 417, 624)
-MatchedPfoId 1, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 1621 (657, 377, 587), nPfoHits 1677 (683, 383, 611)
PrimaryId 2, Nu 1, TB 0, CR 0, MCPDG 2212, Energy 1.89848, Dist. 8.45417, nMCHits 448 (141, 125, 182)
-MatchedPfoId 2, Nu 1 [NuId: 1], TB 0, CR 0, PDG 13, nMatchedHits 407 (122, 116, 169), nPfoHits 444 (124, 131, 189)
PrimaryId 3, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.45388, Dist. 22.4647, nMCHits 442 (102, 198, 142)
-MatchedPfoId 3, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 397 (93, 189, 115), nPfoHits 398 (93, 190, 115)
PrimaryId 4, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.178079, Dist. 30.5587, nMCHits 211 (30, 100, 81)
-MatchedPfoId 4, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 85 (13, 41, 31), nPfoHits 87 (13, 43, 31)

---SUMMARY---
#CorrectNu: 1/1, Fraction: 1
```

Primary true MC target (reconstructable) particles

Interaction type, according to final state target MC particles

Reco PFO matches to each primary, and #hits matched and total



and validate against MC truth

```
lar -c myprotoDUNE_reco.fcl -n 10 my_generated_events.root
```

Now you will be printing this type of information:

```
---INTERPRETED-MATCHING-OUTPUT--- DUNE FD
CCRES_E_P_PIZERO (Nuance 1004, Nu 1, TB 0, CR 0)
IsCorrectNu (NNuMatches: 4)
PrimaryId 1, Nu 1, TB 0, CR 0, MCPDG 11, Energy 1.828, Dist. 43.3081, nMCHits 1712 (671, 417, 624)
-MatchedPfoId 1, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 1621 (657, 377, 587), nPfoHits 1677 (683, 383, 611)
PrimaryId 2, Nu 1, TB 0, CR 0, MCPDG 2212, Energy 1.89848, Dist. 8.45417, nMCHits 448 (141, 125, 182)
-MatchedPfoId 2, Nu 1 [NuId: 1], TB 0, CR 0, PDG 13, nMatchedHits 407 (122, 116, 169), nPfoHits 444 (124, 131, 189)
PrimaryId 3, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.45388, Dist. 22.4647, nMCHits 442 (102, 198, 142)
-MatchedPfoId 3, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 397 (93, 189, 115), nPfoHits 398 (93, 190, 115)
PrimaryId 4, Nu 1, TB 0, CR 0, MCPDG 22, Energy 0.178079, Dist. 30.5587, nMCHits 211 (30, 100, 81)
-MatchedPfoId 4, Nu 1 [NuId: 1], TB 0, CR 0, PDG 11, nMatchedHits 85 (13, 41, 31), nPfoHits 87 (13, 43, 31)

---SUMMARY---
#CorrectNu: 1/1, Fraction: 1
```

Primary true MC target (reconstructable) particles

Interaction type, according to final state target MC particles

Reco PFO matches to each primary, and #hits matched and total

Summary: was the event correctly reconstructed?



When/how do we say an event was correct?

We are quite strict and require for an event to be deemed correct:

Exactly one reconstructed particle must be matched to each target (reconstructable) MC particle, including correct parent-daughter links

E.g.

